An Executable Model of a Simple Genetic Algorithm

Darrell Whitley
Computer Science Department
Colorado State University
Fort Collins, CO 80523
whitley@cs.colostate.edu

Abstract

A set of executable equations are defined which model the ideal behavior of a simple genetic algorithm. The equations assume an infinitely large population and require the enumeration of all points in the search space. When implemented as a computer program, the equations can be used to study problems of up to approximately 15 bits. These equations represent an extension of earlier work by Bridges and Goldberg. At the same time these equations are a special case of a model introduced by Vose and Liepins. The various models are reviewed and the predictive behavior of the executable equations is examined. Then the executable equations are extended by quantifying the behavior of a reduced surrogate operator and a uniform crossover operator. In addition, these equations are used to study the computational behavior of a parallel island model implementation of the simple genetic algorithm.

1 Introduction

The original schema theorem (Holland, 1975) provides a lower bound on changes in the sampling rate for a hyperplane from generation t to generation $t+1$. The following version of the schema theorem is based on Schaffer's work (1987).

$$P(h,t+1) \geq P(h,t)\,\frac{f(h)}{\bar{f}}\,\left\{1 - P_c\frac{l(h)}{L-1}(1 - P(h,t)\,\frac{f(h)}{\bar{f}})\right\}$$

where h represents a particular hyperplane and $P(h,t)$ indicates the fraction of the population that are members of a hyperplane h at time t, where t is an index to a particular generation of strings. The value $f(h)/\bar{f}$ indicates the average fitness of the members of h relative to average fitness of all strings in the population. The defining length of the schema, $l(h)$, includes only the significant part of the defining schema for h. In other words, the defining length of ****10*1*0***** is 5 since there are 5 potential break points between the bits in the significant part of the schema. The length of the string is represented by L and P_c is the probability that crossover will be applied. Note that no mutation is included in this particular formulation of the schema theorem.

This formula captures the following ideas. Together, P_c and $\{l(h)/L-1\}$ define the probability that crossover will cut the schema representing the hyperplane h. However, this need not always be disruptive if the other parent string also samples the same hyperplane h. The probability that a random string in the population contains a copy of the schema representing h is $P(h,t)$ so $1-P(h,t)$ is the probability that the mate *does not* sample the schema representing h. Therefore, the probability of disruption is (1) the probability of crossover multiplied by (2) the probability that crossover will occur within the defining length of the schema multiplied by (3) the probability the other parent is not a member of $P(h,t)$. The probability that a schema survives intact is 1 minus the probability of disruption. Note that the equation is an inequality because it ignores string gains and overestimates the sources of disruption. The sources of disruption are over estimated because recombination with strings containing "similar schemata" (i.e., strings that have many of the relevant bits in common with h, but not all the bits are identical) do not necessarily produce disruption. *String gains* occur when new schema samples are generated by recombination of strings that do not contain copies of the schema h.

The following equations have recently been used to study 3 and 4 bit problems (Whitley, 1992). These equations are based on similar equations for 2 bit problems used by Goldberg (1987) to study the minimally deceptive problem.

$$P(Z,t+1) = (1-P_c)P(Z,t)\frac{f(Z)}{\bar{f}} + P_c\left\{P(Z,t)\frac{f(Z)}{\bar{f}}(1-\text{losses}) + \text{gains.}\right\}$$

which reduces to $\quad P(Z,t+1) = P(Z,t)\frac{f(Z)}{\bar{f}}(1-\{P_c \text{ losses}\})+\{P_c \text{ gains.}\}$

This is an idealized version of a simple genetic algorithm (Goldberg, 1989a). In the current formulation, Z might refer to either a string or a schema representing a hyperplane. P(Z,t) is the proportion of the population that samples the string (schema) Z at time t. The expression $f(Z)/\bar{f}$ indicates the change in representation of Z in the population due to fitness, where f(Z) is the fitness associated with the string (schema) Z and $\bar{f}$ is the average fitness of the population. Given a specification of Z, one can calculate string losses and gains. Losses occur when a string (schema) crosses with another string and the resulting offspring fails to preserve the original string. Gains occur when two different strings cross and independently create a new copy of some string. For example, if Z = 000 then recombining with 100 or 001 will always produce a new copy of 000. Assuming 1-point crossover is used as an operator, the probability of "losses" and "gains" for the string Z = 000 are calculated as follows:

$$\text{losses} = P_{I0}\frac{f(111)}{\bar{f}}P(111,t) + P_{I0}\frac{f(101)}{\bar{f}}P(101,t)$$
$$+P_{I1}\frac{f(110)}{\bar{f}}P(110,t) + P_{I2}\frac{f(011,t)}{\bar{f}}P(011,t).$$

$$\text{gains} = P_{I0}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(100)}{\bar{f}}P(100,t) + P_{I1}\frac{f(010)}{\bar{f}}P(010,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+P_{I1}\frac{f(011)}{\bar{f}}P(011,t)\frac{f(100)}{\bar{f}}P(100,t) + P_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(110)}{\bar{f}}P(110,t)$$
$$+P_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(010)}{\bar{f}}P(010,t).$$

When Z is a string $P_{I0} = 1$ and represents the probably of crossover somewhere on the string. (If Z were a schema, then this term would be the same as $l(h)/(L-1)$) where $l(s)$ is the defining length of the schema representing hyperplane h and L is the length of the string.) For Z = 000 this can be further broken into two order-2 schemata that are of interest. The first order-2 schema of interest, h_1, spans from the first to the second bit and the second order-2 schema of interest, h_2, spans from the second to the third bit. These are of interest because the defining lengths of these order-2 schemata provide probability information about the disruption of the order-3 schema. The probability that one-point crossover will fall between the first and second bit is merely $l(h_1)/(L-1)$, which will be referred to as P_{I1}. Likewise, P_{I2} will denote the probability that one-point crossover will fall between the second and third bit which is given by $l(h_2)/(L-1)$.

The equations can be generalized to cover the remaining 7 strings in the space by using $(S_i \oplus Z)$ to transform each bit string, S_i, contained in the formula to the appropriate corresponding strings for computing the formula for any and all terms of the form P(Z,t+1). The translation is accomplished using bitwise addition modulo 2 (i.e., a bitwise exclusive-or denoted by $\oplus$) (c.f., Liepins and Vose, 1990).

The equations already presented are applicable to 3 bit problems corresponding to either schemata or strings of 3 bits; of course, if schemata are modeled (as opposed to complete strings) the equations would not include information about higher order schemata competitions or the influence of "co-lateral" schemata competitions on the search process. It would also be difficult to obtain fitness information about competing sets of hyperplanes. Therefore in the remainder of this paper, the equations are developed only for strings.

2 A Generalized Form Based on Equation Generators

The 3 bit equations were generated by hand; however, a general pattern exists which allows the automatic generation of equations for arbitrary problems (Whitley, Das and Crabb, 1992). The number of terms in the equations is greater than the number of strings in the search space. Therefore it is only practical to develop exact equations for problems with approximately 15 bits in the encoding.

The development of a general form for these equations is illustrated by generating the loss and gain terms in a systematic fashion. Note that the equations need only be defined once for one string in the space; the *standard form* of the equation is always defined for the string composed of all zero bits. Let S represent the set of binary strings of length L, indexed by i. In general, the string composed of all zero bits is denoted S_0.

All the information needed to define the equations comes from the schema theorem except for the calculation of exact gains and losses. In the process of developing the general pattern for these gains and losses it was realized that the "generators" developed for these executable equations are equivalent to those used by Bridges and Goldberg (1987). The "generators" of Bridges and Goldberg will be reviewed and a general graph structure is introduced to allow better visualization of the relationship between generators.

2.1 Generating String Losses for 1-point crossover

Bridges and Goldberg (1987) present a "middle function" which is used to calculate losses dues to crossover disruption. Consider two strings 00000000000 and 00010000100. Using 1-point crossover, if the crossover occurs before the first "1" bit or after the last "1" bit, no disruption will occur. Any crossover between the 1 bits, however, will produce disruption: neither parent will survive crossover. Also note that recombining 00000000000 with any string of the form 0001####100 will produce the same pattern of disruption (Bridge and Goldberg denoted this as 0001****100; # is used here instead of * to better distinguish between a generator and the corresponding hyperplane.) Bridges and Goldberg formalize the notion of a generator in the following way. Consider strings B and B′ where the first x bits are equal, the middle $(\delta + 1)$ bits have the pattern $b\#\#...\#b$ for B and $\bar{b}\#\#...\#\bar{b}$ for B′. Given that the strings are of length L, the last $(L - \delta - x - 1)$ bits are equivalent. The $\bar{b}$ bits are referred to as "sentry bits" and they are used to define the probability of disruption. The full generator for a middle function M[B,δ,x] is defined as follows:

$$M[B, \delta, x] = b_0 \ ... \ b_{x-1}\bar{b}_x \ ... \ \bar{b}_{x+\delta}b_{x+\delta+1} \ ... \ b_{l-1}$$

or, for the standard form of the executable equations,

$$M[S_0, \delta, x] = 0_0 \ ... \ 0_{x-1}1_x \ ... \ 1_{x+\delta}0_{x+\delta+1} \ ... \ 0_{l-1}$$

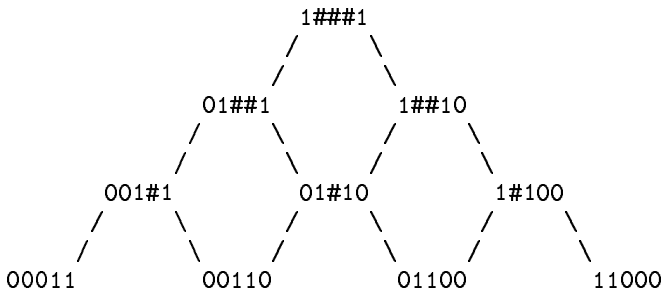
since $B = S_0$ and therefore the sentry bits must be 1.

The middle function generators are then used to count all strings that are capable of disruption as well as the probability of disruption which is derived from the δ length (analogous to the schema defining length) of each particular generator. The probability of disruption for B given by Bridges and Goldberg (1987) is:

$$P_d(B) = \left[\sum_{\delta=1}^{L-1} \frac{\delta}{L-1} \cdot \sum_{x=1}^{L-\delta-1} \sum_{\{j \,|\, S_j \epsilon M[B,\delta,x]\}} {R_j}^t \right]$$

where ${R_j}^t = {P_b}^t(f_b/\bar{f})$ and ${P_b}^t$ is the representation of B in the population at time t and $f_b/\bar{f}$ is its change in representation due to fitness.

The following directed acyclic graph illustrates all generators for "string losses" for the standard form of a 5 bit equation for S_0.



The graph structure allows one to visualize the set of all generators for string losses. In general, the root of this graph is defined by a string with a sentry bit in the first and last bit positions, and the generator token "#" in all other intermediate positions. A move down and to the left in the graph causes the leftmost sentry bit to be shifted right; a move down and to the right causes the rightmost sentry bit to be shifted left. Since the standard form of the equation is written for S_0, the sentry bits are always "1" bits. All bits outside the sentry positions are "0" bits. Summing over the graph, one can see that there are $\sum_{j=1}^{L-1} \; j \cdot 2^{L-j-1}$ or $(2^L - L - 1)$ strings generated as potential sources of string loss.

Note that there are $(2^L - L - 1)$ strings that cause disruption and that the term $\delta/(L-1)$ used in the Bridges and Goldberg equations corresponds to the disruption probability for each string. This term is the same as the continuous "interiors" used to define the 3 bit equations. These generators are used to define all string loss source terms for the executable equations and the "interior" disruption probabilities. For each string S_i produced by one of the "middle" generators in the above graph structure, a term of the following form is added to the *losses* equations:

$$\delta(S_i)\frac{f(S_i)}{\bar{f}} \; P(S_i, t)$$

where $\delta(S_i)$ is a function that calculates the number of crossover points between sentry bits in string S_i, thus defining the critical "interior" region for this particular string. This function does not explicitly require S_0 as an input, since its form is implicitly given by any S_i. This notation attempts to be consistent with Bridges and Goldberg's use of δ. Note, however, that $\delta(S_i)$ does not refer to the *defining length* of a schema (as is often the case in the genetic algorithm literature).

2.2 Generating String Gains for 1-point crossover

Bridges and Goldberg note that string gains for a string B are produced from two strings Q and R which have the following relationship to B.

Region ->	beginning	middle	end
Length ->	a	r	w
Q Characteristics	$\#\#...\#\bar{b}$	=	=
R Characteristics	=	=	$\bar{b}\#...\#$

The "=" symbol denotes regions where the bits in Q and R match those in B; again B = S_0 for the *standard form* of the equations. Sentry bits are located such that 1-point crossover between sentry bits produces a new copy of B, while crossover of Q and R outside the sentry bits will not produce a new copy of B.

Bridges and Goldberg define a beginning function A[B,α] and ending function $\Omega[B,\omega]$, assuming $L-\omega > \alpha - 1$, where

$$A[B,\alpha] = \#\#...\#\#\bar{b}_{\alpha-1}b_{\alpha}...b_{L-1}$$

and $$\Omega[B,\omega] = b_0...b_{L-\omega-1}\bar{b}_{L-\omega}\#\#...\#\#$$

or, for the standard form of the equations:

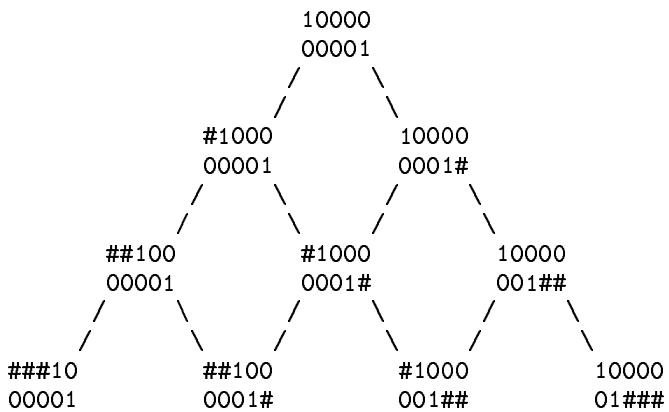
$$A[S_0,\alpha] = \#\#...\#\#1_{\alpha-1}0_{\alpha}...0_{l-1}$$

and $$\Omega[S_0,\omega] = 0_0...0_{L-\omega-1}1_{L-\omega}\#\#...\#\#.$$

The length of the "middle" overlapping region between sentry bits in generators A and Ω (as well as any pair of resulting strings Q and R) is denoted by σ. Bridges and Goldberg use the generators to calculate the probability of string gains for some string B:

$$P_g(B) = \frac{1}{2}\begin{pmatrix} 2 \\ 1 \end{pmatrix} P_c \cdot \sum_{\alpha=1}^{L-1}\sum_{\omega=1}^{L-\alpha}\frac{\sigma+1}{L-1}\left[\sum_{\{j\,|S_j \epsilon A[B,\alpha]\}} {R_j}^t \left[\sum_{\{k|S_k \epsilon \Omega[B,\omega]\}} {R_k}^t\right]\right]$$

These generators can again be presented as a directed acyclic graph structure composed of paired templates which will be referred to as the upper A-generator and lower Ω-generator. The following are the generators in a 5 bit problem.



In this case, the root of the directed acyclic graph is defined by starting with the most specific generator pair. The A-generator of the root has a "1" bit as the sentry bit in the first position, and all other bits are "0." The Ω-generator of the root has a "1" bit as the sentry bit in the last position, and all other bits are "0." A move down and left in the graph is produced by shifting the left sentry bit of the current upper A-generator to the right. A move down and right is produced by shifting the right sentry bit of the current lower Ω-generator to the left. Each vacant bit position outside of the sentry bits which results from a shift operation is filled using the # symbol.

This graph structure allows one to observe that the number of generator pairs at each level of the tree is equal to the depth of the level at which the pair is located (where the root is level 1). Also, for any level k, the number of string pairs generated at that level is 2^{k-1} for each pair of generators. Therefore, the total number of string pairs that must be included in the equations to calculate string gains is:

$$\sum_{k=1}^{L-1} k \cdot 2^{k-1}$$

where L is the length of the string. When expressed in graph form all generators in a particular level in the graph have the same critical defining length between sentry bits *when modeling binary strings*; if the equations are used to model schemata however, the different "interiors" between sentry bits associated with generators at any one level in the tree may have different defining lengths.

Bridges and Goldberg use the equations for $P_d(B)$ and $P_g(B)$ to directly write a formula for the expected proportion of string B in generation t+1 under recombination and crossover with the generators for losses and gains included in the formula. The main difference in the approach offered in this paper is that the generators are used as part of a two stage computation where the generators are first used to create an exact equation in standard form instead of being included directly in the equation for some arbitrary string H. This allows the resulting equations to be implemented in an executable form with a simple transformation function for all other strings in the space.

The probability of a string gain is calculated using σ; since σ is based on the location of sentry bits, each generator pair has its own σ value which is associated with all string pairs produced by that particular generator. All possible string pairs are produced using the generators. Let $S_{\alpha+x}$ and $S_{\omega+y}$ be two strings produced by a generator pair, such that $S_{\alpha+x}$ was produced by the A-generator and has a sentry bit at location $\alpha - 1$ and $S_{\omega+y}$ was produced by the Ω-generator with a sentry bit at $L - \omega$. (The x and y terms are simply correction factors added to α and ω in order to uniquely index a string in S.) Let the σ associated with $S_{\alpha+x}$ and $S_{\omega+y}$ be computed by a corresponding function $\sigma(S_{\alpha+x}, S_{\omega+y}) = L - \omega - \alpha$. For each string pair $S_{\alpha+x}$ and $S_{\omega+y}$ a term of the following form is added to the *gains* equations:

$$\frac{\sigma(S_{\alpha+x}, S_{\omega+y}) + 1}{L-1} \; \frac{f(S_{\alpha+x})}{\bar{f}} \; P(S_{\alpha+x}, t) \; \frac{f(S_{\omega+y})}{\bar{f}} \; P(S_{\omega+y}, t)$$

where $(\sigma(S_{\alpha+x}, S_{\omega+y}) + 1)/(L-1)$ is the probability that 1-point crossover will fall within the "interior" region defined by the sentry bits located at $\alpha - 1$ and $L - \omega$.

3 The Vose and Liepins Models

The executable equations represent a special case of the model of a simple genetic algorithm introduced by Vose and Liepins (1991). In the following discussion the relationship between the standard form of the executable equations presented here and the Vose and Liepins model is examined.

In the Vose and Liepins model, the vector $s^t \in \Re$ represents the t th generation of the genetic algorithm and the i th component of s^t is the probability that the string i is selected for the gene pool. Using i to refer to a string in s can sometimes be confusing. The symbol S has already been used to denote the set of binary strings, also indexed by i (in this case, the numeric value of i); this notation will be used where appropriate to avoid confusion. Note that s^t corresponds to the expected distribution of strings in what is sometimes referred to as the "intermediate population" in the generational reproduction process (after selection has occurred, but before recombination).

In the Vose and Liepins formulation,

$$s_i^t \sim P(S_i, t) f(S_i)$$

where $\sim$ is the equivalence relation such that $x \sim y$ if and only if $\exists \lambda > 0 \mid x = \lambda y$. In this formulation, the term $1/\bar{f}$, which would represent the average population fitness normally associated with fitness proportional reproduction, can be absorbed by the λ term.

The expected number of strings is also represented as a vector $p^t \in \Re$ where the k th component of the vector is equal to the proportional representation of k at generation t **before** selection occurs. This would be the same as $P(S_k, t)$ in the notation more commonly associated with the schema theorem. Finally let $r_{i,j}(k)$ be the probability that k results from the recombination of strings i and j. Now, using $\mathcal{E}$ to denote expectation,

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

To further generalize this model, the function $r_{i,j}(k)$ is used to construct a mixing matrix M where the i,jth entry $m_{i,j} = r_{i,j}(0)$. Note that this matrix gives the probabilities that crossing strings i and j will produce the string S_0. For current purposes, assume *no mutation* is used and 1-point crossover is used as the recombination operator. Then matrix M is obviously symmetric and is zero everywhere on the diagonal except for entry $m_{0,0}$ which is 1.0. Note that M is expressed entirely in terms of string gain information. Therefore, the first row/column of the matrix is the *inverse* of that portion of the string *losses* probabilities that corresponds to $\delta(S_i)$, where each string in the set S is crossed with S_0. For completeness, the $\delta(S_i)$ for strings not produced by the string generators is 0.0 and, thus, the probability of obtaining S_0 during reproduction is 1.0. The remainder of the matrix can be calculated using string gain generators and

$$\frac{\sigma(S_{\alpha+x}, S_{\omega+y}) + 1}{L - 1}.$$

For each pair of strings produced by the string gains generators determine their index and enter the value returned by the function into the corresponding location in M. For completeness, $\sigma(S_j, S_k) = -1$ for all arbitrary pairs of strings not generated by the string gains generators, which implies entry $m_{j,k} = 0$.

Once defined M does not change since it is not affected by variations in fitness or proportional representation in the population. Thus, given the assumption of no mutations, that λ is updated each generation to correct for changes in the population average, and that 1-point crossover is used, then the standard form of the executable equations corresponds to the following portion of the Liepins and Vose model:

$$s^T M s$$

where T denotes transpose.

While the fitness terms are defined explicitly in the executable equation, so far it has been assumed that s includes fitness information without explicitly indicating how this is calculated. Fitness information is later added to the Vose and Liepins model. For now, however, this expression is first generalized to cover all strings in the search space. Vose and Liepins formalize the notion that bitwise exclusive-or can be used to remap all the strings in the search space, in this case represented by the vector s. They show that if recombination is a combination of crossover and mutation then

$$r_{i,j}(k \oplus q) = r_{i \oplus k, j \oplus k}(q).$$

This allows one to reorder the elements in s with respect to any particular point in the space. This reordering is equivalent to remapping the variables in the executable equations. A permutation function, ρ, (denoted by σ in Vose and Liepins' (1991) original work) is defined as follows:

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

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

$$\mathcal{M}(s) = < (\rho_0\ s)^T M \rho_0\ s, ..., (\rho_{N-1}\ s)^T M \rho_{N-1}\ s >^T$$

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 is now explicitly introduced to transform the population at the beginning of iteration $t+1$ to the next intermediate population. 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. (In the simplest case the fitness function may directly be the objective function, but it could also be some function of the value returned by the objective function.)

The transformation from the vector p^{t+1} to the next intermediate population represented by s^{t+1} is given as follows:

$$s^{t+1} \sim Fp^{t+1}$$

or, in terms of $\mathcal{M}$,

$$s^{t+1} \sim F\mathcal{M}(s^t).$$

Vose and Liepins give equations for calculating the mixing matrix M which not only includes probabilities for 1-point crossover, but also mutation. The executable model does not extend their work, but the introduction of the directed acyclic graphs does help in the visionalization of string losses and gains and the executable model has been related to earlier work by Bridges and Goldberg. The executable equations are also expressed in terms more commonly used in the genetic algorithm literature. Because the matrices F and M are sparse, the Vose and Liepins models also have the same general complexity as the executable equations.

In the following sections are some simple extensions of the executable models to include other recombination operators as well as preliminary work using these equations to study parallel island model genetic algorithms. More complex extension of the Vose and Liepins model include finite population models using Markov Models (Nix and Vose, 1992). Vose (1992) surveys the current state of this research.

4 Implementation Complexity and Preliminary Results

A computer program has been written which requires access to a copy of an evaluation function, the length of the strings to be processed and the crossover operator to be employeed. The program then generates the standard equation for the string S_0 of appropriate length. The strings that are referenced in the standard equation are generated as variables and are redefined at each iteration using $(S_i \oplus Z)$ to remap all strings S_i. One generation is produced by iterating through all strings in the space and calculating P(Z,t+1) for each string. String distributions are updated after each generation.

The following complexity results pertain to the executable equations using 1-point crossover. Note that the complexity of each version of these equations is dominated by the string gains: $\sum_{j=1}^{L-1} j \cdot 2^{j-1}$. The total number of string loss terms $= 2^L - L - 1$. Also, for each of the 2^L strings representing variables the equations must be redefined at a cost of L per variable. Thus the computation required to remap and execute one equation must be repeated 2^L times per generation yielding a total computation cost on the order of:

$$2^L \left[\sum_{j=1}^{L-1} j \cdot 2^{j-1} + (2^L - L - 1) + (2^L \cdot L) \right]$$

which reduces to

$$2^L \cdot L \left[2^L + 2^{L-1} - L \right].$$

This expression grows extremely fast. For 10 bit problems the equations can be executed for 15 generations in approximately 5 minutes on a 25 MIPS workstation.

5 Other Operators and Computational Behavior

5.1 Reduced Surrogate 1-point Crossover

A reduced surrogate crossover operator reduces parent strings to a skeletal form in which only those bits that differ in the two parents are represented. Thus the strings have corresponding reduced surrogates as follows:

```
001011101101001 ==>  -0--11---1--0-1
011000101001100 ==>  -1--00---0--1-0
```

Recombination is now limited so that only one possible recombination point exists between the bits in the reduced surrogates. Thus, in the strings illustrated here, there are only 5 possible points of recombination for 1-point crossover; there are 6 possible points of recombination if the strings are treated as structures that wrap-around, as is the case in 2-point crossover.

There are several attributes of the reduced surrogate form that makes this an interesting operator for recombination. First, if at least 1 crossover point occurs between the first and last bits in the reduced surrogate, then the offspring will never duplicate the parents. Second, it causes the recombination process to equally weight the probability of generating each offspring which can potentially be produced by a particular operator. Note that 1-point crossover in any continuous region of matching bits in the parents produces exactly the same offspring. Thus, some offspring are more likely to be produced by crossover than others when recombination operators are applied to the non-reduced forms. The reduced surrogate removes this potential source of bias. The calculation for string losses and gains for the reduced surrogate form of 1-point crossover is also simple compared to other operators.

The same sets of strings and paired strings that can produce string losses and string gains for regular 1-point crossover are also sources for the reduced surrogate 1-point crossover. However, the probabilities associated with the losses and gains do change. Assume crossover must occur between the first and last bits in the reduced surrogates unless there is only 1 bit different between parents. This means that *all potential sources of loss will produce a loss with a probably of 1.0,* since recombination on the reduced surrogate forms forces crossover to occur between the sentry bits of strings produced by the *string losses* generators. In general, if only 1 crossover point is allowed at each possible crossover site in the reduced surrogate forms, then all potential sources of loss will produce a loss with probability of 1.0 regardless of whether 1-point, 2-point or n-point crossover is used.

The probabilities for gains are also simple when the reduced surrogate form is used, although the probabilities do not automatically generalize to an arbitrary number of crossover points. The probabilities of a string gain from a pair of strings produced by the 1-point crossover string generators is $1/(H-1)$, where H is the Hamming distance between the two parents. The sentry bits in the string gain generators are effectively adjacent bits in the reduced surrogate forms, so there is only 1 crossover point between the sentry bits. Since there are H bits different in the reduced surrogate form, there are $H-1$ possible crossover points, only 1 of which results in a string gain.

5.2 String Gains and Losses for Uniform Crossover

One can also calculate string gains and losses for the uniform crossover operator. For string losses, one must enumerate all strings in the space that contain 2 or more 1 bits (i.e., all strings that are more than 2 bits different from S_0). The Hamming distance, H, between parents is again critical, since one offspring must inherit all zeros if a loss is to be avoided. The probability of losses is $(1 - 0.5^{H-1})$. The exponent is $H-1$ instead of H since it does not matter which offspring inherits the first 0 bit, but the next $H-1$ bits are critical.

The probability of gains is more complicated and it is again useful to introduce a set of generators. Clearly a zero bit must occur in every position in one of the two parents before it is possible to generate S_0 as an offspring. Thus, for every position in the string, either a zero occurs in both parents or complementary bits occur in both parents. This suggests a general mechanism for calculating string gains.

First, there are $\left(\begin{array}{c} L \\ k \end{array} \right)$ ways to select positions in which both strings have k zeros. All of the cases for $k = 0$ to $L-2$ must be examined. If $k = L$, both parents are S_0 and if $k = L-1$, then exactly one of the parents are S_0; in both of these cases, computing string *gains* is not relevant.

It has already been noted that for those positions where both bits are not zero, the bits must be complements. Consider the case where L = 4 and k = 0. A parent string and the complementary parent string are represented as follows:

$$\begin{aligned} \text{B} &= b_1 b_2 b_3 b_4 \\ \text{C} &= c_1 c_2 c_3 c_4 \end{aligned}$$

This provides a generator, except B cannot = S_0, so we start enumerating B at 1. Furthermore, only half of this space must be enumerated since this is sufficient to enumerate all possible pairs of strings. (While B is enumerated starting at 1 and counting up, the complement is simultaneously enumerated starting at $2^L - 2$ and is decremented.) Fixing the value of the first bit in B which has a complement in C cuts the space in half. Thus, the generator for k=0 is as follows:

$$\begin{aligned} \text{B} &= 0_b b_2 b_3 b_4 \\ \text{C} &= 1_c c_2 c_3 c_4 \end{aligned}$$

The bits 0_b and 1_c are marked for purposes of illustration to indicate they are part of the *bit and complement* portion of the string generator.

In general then, for $k = 0$ to $L-2$ the following process is repeated for each possible assignment of the $\left(\begin{array}{c} L \\ k \end{array} \right)$ ways to select positions in which both strings have zero bits. First the k zero bits are assigned a position, then the remaining bits positions are assigned the symbol "b" (or "c" in the complement) as a place holder. Next the rightmost "b" and "c" is converted to 0 and 1 respectively, which is denoted here by 0_b and 1_c. This produces the set of generators, each indexed by k, which indicates the number of common zero bits. Note that k is directly related to Hamming distance ($L - k = H$) and all pairs of strings produced by a particular

generator are at exactly the same Hamming distance from one another. From each generator the strings are produced by enumerating over the $L-k-1$ "b" bits for B from 1 to $2^{L-k-1}-1$ while generating the corresponding complements bits for C. The complete set of generators for a 4 bit problem is as follows:

For $\begin{pmatrix} 4 \\ 0 \end{pmatrix}$

$0_b b_2 b_3 b_4$
$1_c c_2 c_3 c_4$

For $\begin{pmatrix} 4 \\ 1 \end{pmatrix}$

$00_b b_3 b_4 \quad 0_b 0 b_3 b_4 \quad 0_b b_2 0 b_4 \quad 0_b b_2 b_3 0$
$01_c c_3 c_4 \quad 1_c 0 c_3 c_4 \quad 1_c c_2 0 c_4 \quad 1_c c_2 c_4 0$

For $\begin{pmatrix} 4 \\ 2 \end{pmatrix}$

$000_b b_4 \quad 00_b 0 b_4 \quad 00_b b_3 0 \quad 0_b 00 b_4 \quad 0_b 0 b_3 0 \quad 0_b b_2 00$
$001_c c_4 \quad 01_c 0 c_4 \quad 01_c c_3 0 \quad 1_c 00 c_4 \quad 1_c 0 c_3 0 \quad 1_c c_2 00$

Having generated all possible sources of string gains for uniform crossover, for any string pairs produced by a generator indexed by k the probability of generating the string S_0 is 0.5^{L-k-1}, or 0.5^{H-1} since $L-k=H$.

5.3 Crossover Probability Versus Schema Length and Fully Deceptive Functions

Recall that the crossover probability affects only the term for losses and gains in the following equation:

$$P(Z,t+1) = P(Z,t)\frac{f(Z)}{\bar{f}}(1-\{P_c \text{ losses}\}) + \{P_c \text{ gains.}\}$$

Examination of the executable equations reveals that lowering the crossover probability from 1.0 to 0.5 is exactly the same in these equations as reducing the length of the crossover interiors by exactly half. The assumption of an infinitely large population is critical here. In actual finite populations the effect is not identical, since lowering P_c to 0.5 produces increased duplicates and acts to lower genetic diversity. It also exaggerates biases due to sampling error, since it tends to reinforce any sampling bias that exists in the current population sample. However, given the relationship between P_c and the crossover interiors, it is reasonable to infer that if P_c is sufficiently low, the defining length of critical schemata will *appear* to be shorter than they actually are. Figure 1 illustrate the execution of these equations on an embedded 3 bit fully deceptive problem (c.f., Goldberg, 1989b; Whitley, 1991) as well as the results of executing an actual simple genetic algorithm using a population size of 625. The three bits corresponding to the deceptive function are located at positions 1, 5 and 9; the remaining bits contribute nothing to the evaluation function. When P_c is 0.4 or above, the equations show convergence to the deceptive attractor 000 (Whitley, 1991), while the global optimal solution is at 111. However, when P_c is lowered to 0.35 or below, the equations show that a simple genetic algorithm should correctly converge to the global solution 111 (Whitley, Das and Crabb, 1992). The actual behavior *for a single run* of the simple genetic algorithm is closely predicted by the equations. Note that in order to overcome deception in this way the population must be sufficiently large to prevent the loss of genetic diversity which will result from a low value of P_c.

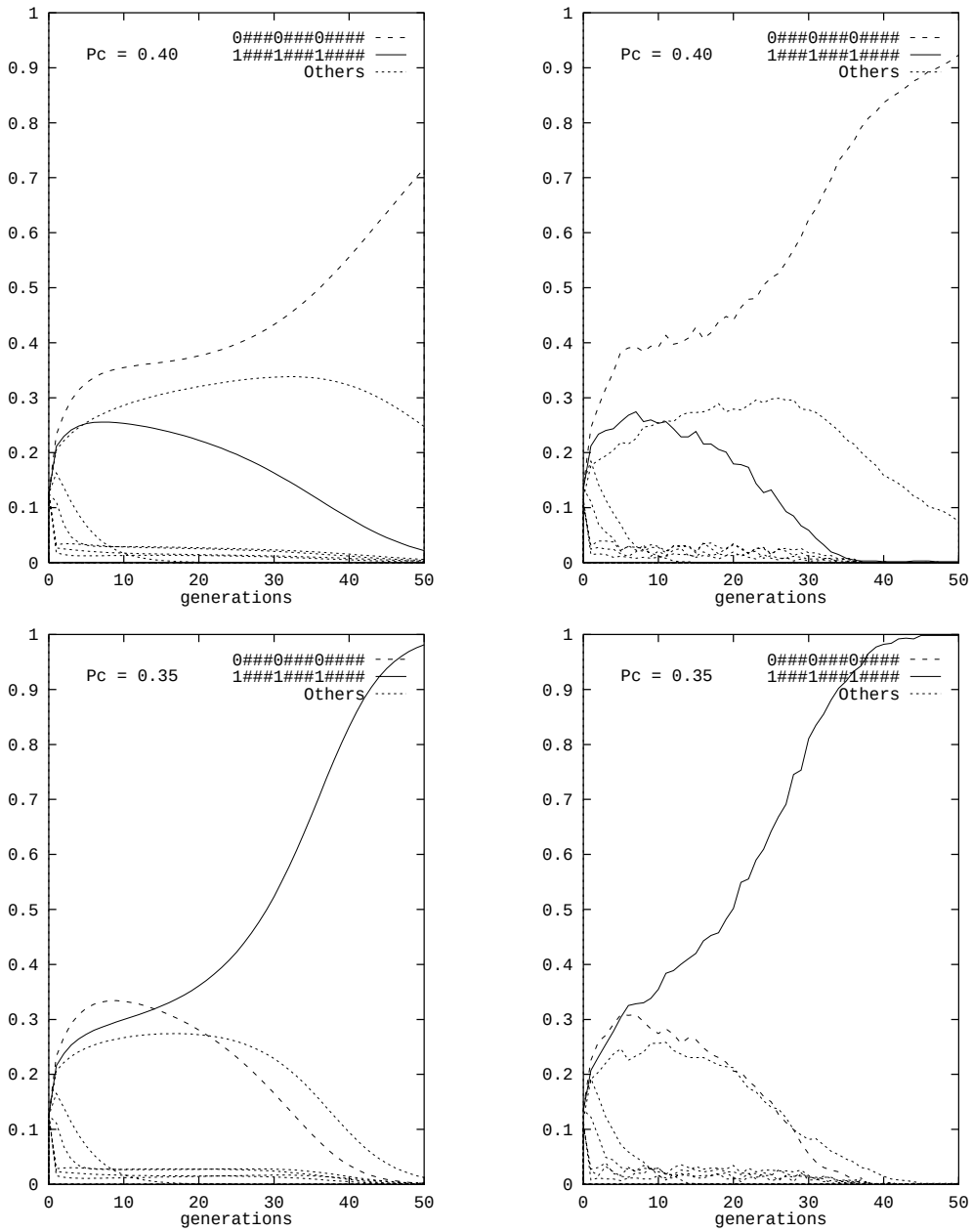


Figure 1: The left hand column shows the results of executing the equations for a fully deceptive three bit subproblem while the right hand column shows the results of executing an actual genetic algorithm (population size 625.) The effects of lowering the probability of crossover, P_c, to 0.40 and 0.35 are clearly illustrated.

5.4 Parallel Island Models

The equations presented in this paper are used to better understand the computational behavior of a parallel island model implementation of a simple genetic algorithm. The island model assumes that the total population used by the genetic algorithm is broken down into a number of subpopulations referred to as islands. Each island is in effective executing some version of the genetic algorithm on the reduced subpopulation. In the equations used here, each island is executing a separate version of the equations for the standard genetic algorithms using 1-point crossover. *Migration* occurs between subpopulations, usually in some restrictive fashion based on temporal and spatial considerations.

In the models implemented here, 4 subpopulations are modeled. The effects of population size are only partially modeled. The distribution of strings in the subpopulations are initialized by generating distinct finite subpopulations. Infinite population sizes are still assumed, however, after the first initial generation. Thus, the equations only model the effects of having different starting points in the different subpopulations based on finite populations. In the equations, migration occurs every X generations and copies of the individuals that make up the top 5% of the island population are allowed to migrate. Note that this top 5% may all be copies of the same string, or it may be some combination of strings. The island receiving these strings deletes the bottom 5% of its own population to absorb the new strings.

The migration scheme used in these equations assumes that the islands are arranged in a ring, and on the first migration strings move from their current island to their immediate neighbor to the left. Migrations occur between all island simultaneously. On the second migration the islands send copies of the top 5% of their current population to the island which is 2 moves to the left in the ring. In general, the migration destination address is incremented by 1 and moves around the ring. This continues with migrations occurring every X generations until each island has sent one set of strings to every other island (not including itself); then the process is repeated.

One thing that is immediately obvious is that migration of the best strings, or of any set of strings that are above average, acts to increase selection due to fitness. Those strings which migrate from one island to another automatically double their representation. Thus, one effect of the parallel island model is to allocate additional reproductive opportunities to those strings which have a high enough fitness to migrate. This could potentially cause premature convergence in a single population genetic algorithm. However, the equations also suggest that the island model may be especially well suited to handling the additional selective pressure. Although the island model equations are very much restricted by the assumption of infinite populations *after the first generation*, the equations nevertheless show that having different starting points for the individual islands may be sufficient to cause each island to follow a unique search trajectory, even though migration will tend to cause the set of equations representing the various islands to become more similar over time. Modeling finite populations and mutation should further distinguish the computational behavior of each of the individual subpopulations.

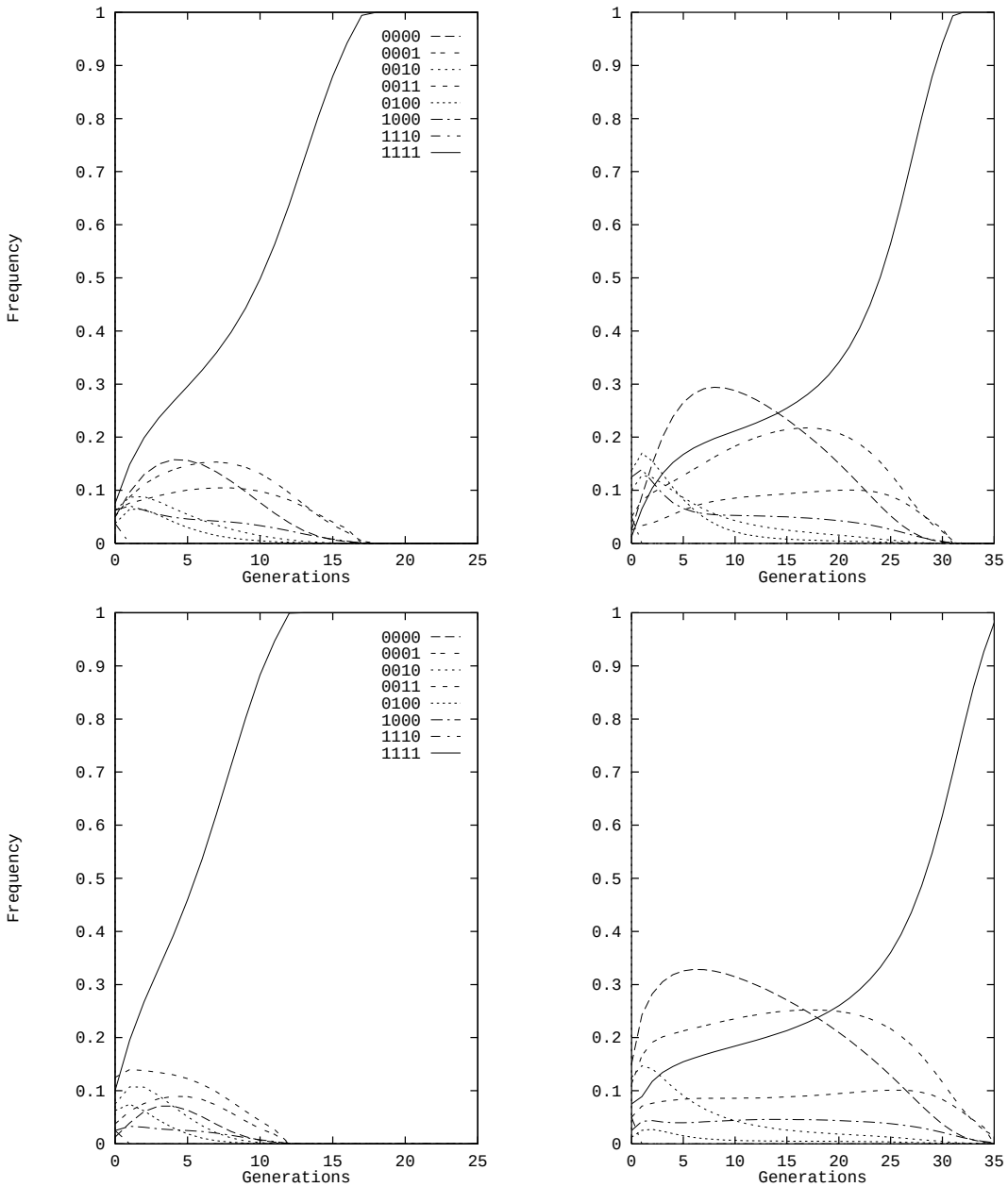


Figure 2: These graphs show the proportional representation of strings in four subpopulations of a Parallel Island Model, where each island executes a simple genetic algorithm. The search behavior of each island is different due to the different initial starting points associated with each island.

Results for a 4 bit fully deceptive problem (Whitley, 1991) are shown in Figure 2. In this figure, strings migrate every generation. The behavior of the models becomes more complex when migration occurs after two or more generations. In this case, every recombination breaks apart the critical order-4 level schemata (corresponding to strings in this case) which are needed to direct the population toward convergence to the global optimum. In the single population model, the genetic algorithm converges to the local optimum 0000; but the equations show that the parallel island model converges to 1111. This is clearly due in part to the higher selective pressure which results from migration but Figure 2 also suggests that the effects of the different search trajectories may be important. If any one subpopulation converges toward the global solution, migration (with its additional selective pressure) can sometimes help to pull the other subpopulations in the same direction.

6 Discussion

Several test problems have recently been constructed to test the optimization abilities of genetic algorithms. These problems are often characterized by the construction and concatenation of smaller functions of 15 bits or less. However, one criticism of these efforts is that the theoretical tools used to construct problems such as deceptive functions are based on static analyses. The models reviewed in this paper introduce a dynamic model of the behavior of a simple genetic algorithm.

The equations can be used to study the computational behavior of the genetic algorithm on small problems. The equations can also be used to study larger function which are constructed by concatenating small functions, since these subproblems typically have no higher-order or co-lateral interaction between hyperplanes (except those that occur due to sampling error and genetic hitch-hiking).

These equations do not include information about sampling error. However, they do offer the potential for obtaining precise information about other aspects of genetic search, such as hyperplane sampling behavior. These equations can also be extended to include various genetic operators and to model other type of genetic search.

Acknowledgements

This research was supported in part by NSF grant IRI-9010546 and by an equipment grant from IBM. An earlier version of the text in section 2 and subsection 5.2 appears in Whitley, Das and Crabb (1992). My thanks to Curtis Crabb for implementing the generalized form of the executable equations and Rajarshi Das for helping to develop the results presented in subsection 5.2.

References

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.*

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

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

Goldberg, D. (1989b) Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems* 3:129-152.

Holland, John (1975) Adaptation In Natural and Artificial Systems. University of Michigan Press.

Liepins, G. and Vose, M. (1990) Representation Issues in Genetic Algorithms. Journal of Experimental and Theoretical Artificial Intelligence, 2:101-115.

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

Schaffer, D. (1987) Some Effects of Selection Procedures on Hyperplane Sampling by Genetic Algorithms. In, *Genetic Algorithms and Simulated Annealing,* L. Davis, ed. Pitman.

Vose, M. (1992) Modeling Simple Genetic Algorithms. *Foundations of Genetic Algorithms -2-,* D. Whitley, ed., Morgan Kaufmann.

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

Whitley, D. (1991) Fundamental Principles of Deception in Genetic Search. *Foundations of Genetic Algorithms.* G. Rawlins, ed. Morgan Kaufmann.

Whitley, D. (1992) Deception, Dominance and Implicit Parallelism in Genetic Search. *Annals of Mathematics and Artificial Intelligence.* 5:49-78.

Whitley, D., R. Das and C. Crabb (1992) Tracking Primary Hyperplane Competitors During Genetic Search. To Appear: *Annals of Mathematics and Artificial Intelligence.*