An Investigation of GA Performance Results for Different Cardinality Alphabets

Jackie Rees and Gary J. Koehler
Decision and Information Sciences
351 BUS
University of Florida
Gainesville, FL 32611

352-392-9600
352-392-5438 (FAX)
jrees@grove.ufl.edu, koehler@nervm.nerdc.ufl.edu

February 6, 1997

ABSTRACT

Theoretical and empirical results give mixed advice for choosing the cardinality for GA representation. Using GA models that capture the exact expected behavior of both the binary and higher cardinality cases, the determination of which representation is best for a given GA can be made. De Jong et al. and Spears and De Jong presented how the exact model for the binary genetic algorithm can give important insights to transient GA behavior. This paper uses a similar approach to study the impact of different cardinalities using the Koehler-Bhattacharyya-Vose general cardinality model.

1. Introduction

In choosing a representation for Genetic Algorithms (GAs), most practitioners use binary alphabets. This is motivated by the Principle of Minimal Alphabets [8] which states that given two possible encodings, the one with the lower cardinality alphabet gives a greater number of schema sampled by a given population. To practitioners, this means selecting the smallest alphabet that permits a natural expression of the problem.

There have been mixed results with different encodings. Some studies have suggested better performance with binary encodings while others have shown higher-cardinality alphabets provide better results [3]. Antonisse [1] argued that higher-cardinality alphabets are theoretically preferable to binary encodings. Since most real world problems do not lend themselves to binary encoding, higher cardinality alphabets might prove to be necessary for these complex problems. Using GA models that capture the exact expected behavior of both the binary case and higher cardinality strings, the determination of which representation is best for a given GA can be made.

In [15], Vose presented an infinite-sized population model for simple binary Genetic Algorithms (GAs) that gives their exact expected behavior over time. Similar models were provided later by T. Davis [4] and Vose and Liepins [17]. This model was extended to finite-sized populations by Nix and Vose in 1992 [12]. However all of these papers restrict GA strings to the binary case.

Some attempts have been made to generalize the Vose models to higher cardinality encodings. Bhattacharyya and Koehler [2] considered GA strings composed of components drawn from 2^v cardinality alphabets and extended the Vose model to cover

this case. Their method of analysis follows the binary case fairly closely, as might be expected. Recently, an exact model for the general cardinality case has been developed by Koehler, Bhattacharyya and Vose [10].

De Jong et al. [6] and Spears and De Jong [14] presented how the exact model for the binary genetic algorithm can give important insights to transient GA behavior. This paper will use a similar approach to study the impact of different cardinalities using the Koehler-Bhattacharyya-Vose general cardinality model.

In Section 2 we provide details on the form of GA model we employ, the general cardinality model that captures the exact expected behavior of these GAs, and formulas for the various waiting-time properties of these models. (De Jong et al. [6] give detailed formulas for many waiting time properties of Markov Chains for Vose binary GA models. In Section 2C we provide matrix-level formulas and cover the general higher-cardinality GA models.)

In Section 3 we discuss our goals and various issues that impact the study of cross-cardinality comparisons of GA performance. In Section 4 we describe our experimental approach. Results are given in Section 5. We discuss these results in Section 6 and give conclusions and future directions in Section 7.

2. Notation and Models

A. Simple GA Algorithm

There are many variants of the simple GA algorithm. For this paper we use the algorithm shown in Table 1. The models given in this paper apply to this algorithm.

Algorithm 1: *(Genetic Algorithm)*

Given:

String length $\ell \geq 2$, cardinality $c \geq 2$, fitness function f(), mutation rate $\mu \in \left(0, 1/c\right)$, one-point crossover rate $\chi \in (0,1]$ and population size $n \geq 1$.

Initialization: Generate an initial population, population 0. This is usually done by randomly drawing n strings from Ω with (or without) replacement.

Step 1: Form a new population as follows. Repeat the following steps until the new population has n members.

(A) Randomly choose two members from the old population according to their fitness values relative to the sum of all fitness values of the current population. These are called parent strings.

(B) Let RAN(0,1) give a uniform random number between zero and one. If RAN(0,1) $\leq \chi$, derive two new strings by a random single-point crossover. Randomly choose one to add to the new population. Otherwise, randomly choose one of the parents to add to the new population.

(C) Perform mutation on the new population member. For each of the ℓ digits, if RAN(0,1) $\leq \mu$ randomly replace the digit by one of the remaining c-1 digits.

Step 2: If stopping conditions are not met, return to Step 1.

Table 1: The Simple Genetic Algorithm with One-Point Crossover and Uniform Mutation

B. Markov Model of Expected GA Behavior

Nix and Vose [12] have derived a Markov Chain (MC) model for the binary GA which directly carries to the higher cardinality case. The MC has states that represent the possible populations. Each state i, i $\in$ {0, ..., N-1}, indexes a vector, ϕ_i, of length $r = c^{\ell}$. Let e be a vector of ones of appropriate length and e' its transpose. The ϕ_i vectors are defined by

$$e'\phi_i = n,$$

$$\left(\phi_i\right)_j \in \{0,1,\ldots,n\} \qquad j = 0, ..., r-1$$

and

$$\phi_i \succ \phi_j \qquad \text{if } i < j.$$

Here $\succ$ means "lexicographically greater than." (Notation used throughout the paper is summarized in Table 2.) Nix and Vose [12] showed that

$$N = \binom{n + r - 1}{r - 1}.$$

Let $Z' = \left(\phi_0, \phi_1, \ldots \phi_{N-1}\right)$ be the matrix of the ϕ vectors and let B be the matrix formed from Z by

$$B_{i,j} = \delta\left(Z_{i,j} \neq 0\right)$$

where

$$\delta(x) = \begin{cases} 0 & x \text{ false} \\ 1 & x \text{ true} \end{cases}.$$

The probability of transition from population i to population j is given by [12]:

$$P_{i,j} = n! \prod_{g=0}^{r-1} \frac{q_{i,g}^{Z_{j,g}}}{Z_{j,g}!}$$

where

$$q_{i,g} = \mathcal{M}\left(\frac{F\phi_i}{e' F\phi_i} \right)_g$$

and F is a diagonal matrix having values f(0), f(1), ..., f(r-1) along the diagonal. Operator $\mathcal{M}()$ of an r dimensional vector is defined by (see [17])

$$\mathcal{M}(x) = \begin{pmatrix} (\sigma_0 x)' M \sigma_0 x \\ . \\ . \\ . \\ (\sigma_{r-1} x)' M \sigma_{r-1} x \end{pmatrix}$$

where the permutation matrix σ_i is defined by $(\sigma_i)_{j,k} = \delta(k = i \oplus j)$. Finally, matrix M is defined by (see [16])

$$M_{y,z} \equiv 0.5 \sum_{p \in \Omega_2} \sum_{j \in \Omega} \mu_j (\chi_p + \chi_{\bar{p}}) [y \otimes p \oplus \bar{p} \otimes z \oplus j = 0]$$

where

$$\chi_p \equiv \begin{cases} \dfrac{\chi}{\ell - 1} & \text{if } p = 2^h - 1 \text{ for some } h \in (0, \ell) \\ 1 - \chi & \text{if } p = 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mu_i \equiv \left(\frac{\mu}{c-1}\right)^{m(i)} (1-\mu)^{\ell - m(i)}.$$

The latter two quantities are crossover and mutation masks (see [16]). Under our assumptions on μ, χ, and f(), the transition matrix, P, is completely ergodic.

C. Expected Waiting Times

We start our GA with an initial population, say population h. The probability that h contains string i is merely $e_h' B e_i$. Either string i is in population h or not. If state h does not contain i, then the probability of first seeing i after one transition is equal to the sum of transition probabilities from state h to states containing string i. Likewise, the probability of first seeing string i starting in state h after t transitions (i.e., after t new generations) is equal to the sum of probabilities of going through t-1 states not containing string i to a state containing string i. Let e_i be the i^{th} unit vector and d(x) be the diagonal matrix formed from any vector x. In matrix notation, the vector of first passage probabilities to string i in t>0 transitions is

$$FP_{i,t} = d(e - Be_i)\left[d(e - Be_i)Pd(e - Be_i)\right]^{t-1} PBe_i. \qquad (1)$$

The first term on the right has zeros on the diagonal for states containing string i and ones otherwise. The second term gives t-1 step transition probabilities through states not containing string i. The final term gives one-step transition probabilities to states containing string i. Clearly $FP_{i,0} = Be_i$. Let $D_i = d(e - Be_i)$. Then Equation 1 can be written more succinctly as

$$FP_{i,t} = \left[D_i P\right]^t Be_i.$$

c	cardinality of string alphabet.
$\oplus$	$a \oplus b = (a+b) \bmod c$. When c is 2, $\oplus$ is the bitwise exclusive-or operator.
$\otimes$	$a \otimes b = (ab) \bmod c$. When c is 2 $\otimes$ is the bitwise and operator.
$\bar{k}$	complement of k defined by $k \oplus \bar{k} = 1$
$m(x)$	number of nonzero digits of string x.
μ	uniform mutation rate with $\mu \in \left(0, 1/c\right)$.
χ	one-point crossover rate with $\chi \in (0,1]$.
μ_i, χ_p	mutation and crossover masks, respectively.
ℓ	string length.
r	number of strings, $r = c^{\ell}$.
Ω	set of all strings, $\Omega = \{0,1,\ldots,r-1\}$. $\Omega_2 = \{0,1,\ldots,2^{\ell}-1\}$
n	population size.
x'	transpose of vector/matrix x.
ϕ_i	r-vector of string frequencies in population i.
N	number of possible populations.
Z	Nxr matrix of ϕ_i vectors. $Z' = [\phi_0, \phi_1, \ldots \phi_{N-1}]$.
B	Nxr matrix formed by setting nonzero components of Z to one.
M	rxr mixing matrix.
$\delta(x)$	is 0 if x is false, 1 otherwise.
σ_i	i^{th} permutation matrix with $(\sigma_i)_{j,k} = \delta(k = i \oplus j)$.
$\mathcal{M}(\ )$	r-vector recombination operator.
$f(i)$	fitness of string i.
F	focusing matrix, a diagonal matrix with $F_{i,i} = f(i)$.
P	transition matrix over state space of populations.
$d(x)$	a diagonal matrix formed using vector x as the diagonal.
e, e_i	vector of ones and i^{th} unit vector, respectively.
D_i	$d(e - Be_i)$.
π	N-vector of prior probabilities over the population states.
$FP_{i,t}$	expected first passage probabilities to string i in exactly t generations.
$SFP_{i,t}$	probability of expected first passage to states containing i in t generations.
ET_t	vector of expected times to see string i.
EWT_t	expected waiting time to see string i.

Table 2: Summary of Notation

Thus, the vector of expected first passage probabilities to states containing string i in exactly t or fewer generations is

$$\begin{aligned} SFP_{i,t} &= \sum_{k=0}^{t} \left[D_i P\right]^k Be_i \,. \\ &= \left(I - \left[D_i P\right]^{t+1}\right)\left(I - \left[D_i P\right]\right)^{-1} Be_i \\ &= \left(I - \left[D_i P\right]^{t+1}\right) e \end{aligned}$$

since

$$\left(I - \left[D_i P\right]\right)^{-1} Be_i = e$$

and

$$\sum_{k=0}^{t} \left[D_i P\right]^k = \left(I - \left[D_i P\right]^{t+1}\right)\left(I - \left[D_i P\right]\right)^{-1} .$$

The expected time till string i is seen can be computed by

$$ET_i = \sum_{t=0}^{\infty} t FP_{i,t} = \sum_{t=0}^{\infty} t\left(D_i P\right)^t Be_i = \left(I - D_i P\right)^{-1} e - e \,.$$

Let π represent the prior probabilities for choosing population zero. Then the probability of expected first passage to states containing i in t generations or less is

$$EFP_{i,t} = \pi' SFP_{i,t} \qquad (2)$$

and the expected waiting time to see string i is

$$EWT_i = \pi' ET_i$$

where π is a vector of prior probabilities of choosing each state for population zero. Since the initial population is typically chosen randomly from Ω with replacement, we can use

$$\pi_j = \frac{n!}{r^n \prod_{g=0}^{r-1} \left(\phi_j\right)_g !}.$$

If the initial population is chosen randomly without replacement we have

$$\pi_j = \frac{\delta\left(\phi_j \le e\right)}{\binom{r}{n}}.$$

3. Goals and Issues

Our overall goal is to explore the theoretical expected performance of a Simple GA using different cardinality representations of the original problem. This objective is made more explicit in the following subsections.

A. Comparing Performance

To determine the effect of cardinality on performance, we will compare the expected waiting time to an optimal string under different cardinality representations of some chosen problems.

Other measures of performance are possible. In practice, one measure often used is the number of populations before an optimal solution first arises. Theoretically, this statistic is completely characterized by the distribution of first passage probabilities to optimal strings. The weighted time to the first appearance of an optimal string using this distribution is just the expected waiting-time measure we propose.

Another approach used in practice is to compare the average population fitnessess of two (or more) approaches. However, this measure doesn't give clear guidelines to performance since these curves seldom dominate each other.

An advantage of using expected waiting time is that this is a scalar measure of a GA's performance and is easy to use in comparisons. In any case, there are several issues we need to address when varying the cardinality of a GA representation and these are covered below.

B. Representation

Given a search problem, the representation of the search strings may not lend itself directly to arbitrary cardinalities. For example, suppose we have an objective function $g(x_1, x_2)$ where $x_i \in \{0,1,...,80\}$. This problem can be naturally represented by using

- $c = 3$ and $\ell = 6$ or by
- $c = 9$ and $\ell = 4$ or by
- $c = 81$ and $\ell = 2$.

(Using $c = 6561$ would not be permitted under our GA since its string length would be 1.) In each case, the original x values can be retrieved from the integer value of a string, s, by

$$x_2 = \left\lfloor \frac{s}{81} \right\rfloor$$

(where $\lfloor x \rfloor$ and $\lceil x \rceil$ are the floor and ceiling of x, respectively) and

$$x_1 = s - 81x_2 .$$

A natural fitness function might be

$$f(s) = g(x_1, x_2) = g\left(s - 81\left\lfloor \frac{s}{81} \right\rfloor, \left\lfloor \frac{s}{81} \right\rfloor \right).$$

This problem cannot be represented "naturally" using binary or any other cardinality. To represent this problem using a binary representation, a number of strategies can be invoked.

One strategy is to penalize the fitness values of infeasible strings. For example, if we use a binary representation of $x_i \in \{0,1,...,80\}$ (say with $\ell = 13$) then 1,631 of the 8,192 possible strings don't conform to any valid string in the original problem range. These could receive a very small fitness value in comparison to valid values of $g(x_1, x_2)$. Possible drawbacks are the distortion in selection pressures brought about by different penalty structures or the different ways that the 1,631 invalid strings can be chosen and their impact on reproduction.

Another strategy is to stretch the original fitness function to cover the whole range of the representation. For example, using a binary representation with $\ell = 14$ given the original ranges of $x_i \in \{0,1,...,80\}$, we might have

$$x_2 = \left\lceil \frac{80}{127} \left\lfloor \frac{s}{128} \right\rfloor \right\rceil$$

and

$$x_1 = \left\lceil \frac{80}{127} \left(s - 128 \left\lfloor \frac{s}{128} \right\rfloor \right) \right\rceil.$$

However, this method has drawbacks. Some (x_1, x_2) points will be mapped into more frequently than others. For example, s = 13 and 14 both map to (0,9) but only s = 15 maps to (0,10).

In this paper we will choose problems that can be naturally represented under different cardinalities.

C. Mutation and Crossover Rates

If we wish to compare the performance impact of using different cardinalities, we need to control the impact of other parameters on the performance measure. Two parameters under our control are the mutation and crossover rates.

It seems reasonable to assume that a mutation rate used at one cardinality may not be appropriate for a different cardinality. For example, say we use a mutation rate of 0.01 for a binary problem of string length 4 and for a problem with cardinality c = 4 and string length 2. The probabilities of mutating to different domain values would be different. For example, the probabilities of mutating to various strings starting from string s having an integer value of 13 (1101 in binary and 31 with c = 4) would be:

Mutate s=13 to	Binary Probability	c = 4 Probability
0	0.00000099	1.11111E-05
1	0.00009801	0.0033
2	0.00000001	1.11111E-05
3	0.00000099	1.11111E-05
4	0.00009801	1.11111E-05
5	0.00970299	0.0033
6	0.00000099	1.11111E-05
7	0.00009801	1.11111E-05
8	0.00009801	0.0033
9	0.00970299	0.9801
10	0.00000099	0.0033
11	0.00009801	0.0033
12	0.00970299	1.11111E-05
13	0.96059601	0.0033
14	0.00009801	1.11111E-05
15	0.00970299	1.11111E-05

There is no simple way to keep these probabilities equal.

Now consider crossover. The crossover rate merely specifies the probability that two strings will mate. The same rate used for one cardinality appears to have the same impact as another cardinality. Any two strings will face the same probability of mating. However, given two strings are to mate, there will be fewer crossover points for the higher cardinality string, which also impacts the overall reproduction probabilities.

In order to balance these factors, we will use a strategy given by De Jong et al. [6] and compare the expected waiting times of the different cardinalities evaluated at the mutation and crossover rates that minimize this value. In other words, we will compare the best realization of expected waiting times.

D. Sampling of the Initial Population

The initial population of a GA is typically drawn randomly from Ω with replacement. However, more diversity (at a slightly higher initial computation cost) can be obtained by sampling without replacement. We will compute expected waiting times under both initial conditions.

E. Multiple Optima

If there are multiple optimal strings, the expected waiting time to any of them can be computed by a simple variation of the formulas give in Section 2C. However, we choose functions having a unique global optimum (with, perhaps, local optima) to push our analysis to the extreme.

F. Computational Limitations

Computing expected waiting times requires computing and using the transition matrix P. Unfortunately, P is N by N and N can be very large. For example, the De Jong test function F1 [5] has three variables each with a range of 1024 values. A binary encoding would require a 30 bit string resulting in

$$N = \binom{n + 2^{30} - 1}{2^{30} - 1}$$

which is over 10^{17}. At this point, practical considerations limit us to exploring problem/population sizes having small r and n values.

4. Experimental Design

A. Hypotheses

The main hypothesis we seek to test is the Principle of Minimal Alphabets. An implication of this principle is the following:

Hypothesis 1: *(Principle of Minimal Alphabets)*

The expected waiting time to see the optimal string should be minimal for the smallest natural cardinality representation.

A slightly stronger hypothesis will also be tested:

Hypothesis 2:

The expected waiting time to see the optimal string should be non-increasing with decreasing cardinality.

We also seek to test a few other hypotheses. Since choosing an initial population by randomly sampling strings without replacement gives more diversity, one might expect faster discovery of an optimal string. This leads to:

Hypothesis 3:

The expected waiting time to see the optimal string under sampling with replacement should not be lower than that obtained with sampling without replacement.

Along similar lines, increasing the population size should also decrease the expected time to see an optimal solution. This gives:

Hypothesis 4:

The expected waiting time to see the optimal string should be non-increasing in increasing population size.

B. Test Bank

We have chosen four test functions to explore (see Table 3.) The first two are restricted versions of De Jong's F1 and F8 functions [5]. The third is a fully deceptive trap function for binary problems [7] and the fourth is a higher cardinality deceptive function [11].

B1. De Jong's F1 (Modified)

De Jong's F1 is simplified to one variable and converted to a maximization form having all positive values. The original function is

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 \qquad -5.12 \le x_i \le 5.11.$$

We first restrict the function to one variable with a range of $100x \in \left[-\frac{r}{2}, \frac{r}{2}\right)$. To convert it to a maximization problem we subtract it from its maximum (plus a small number to keep it positive). Thus we get

$$f(x) = \frac{4 + r^2}{40000} - \frac{x^2}{10000}.$$

Converting this to a function of a string value requires shifting s from a range of $s \in [0, r)$ to x's range. Thus, our modified F1 function is

$$f(x) = \frac{4 + r^2}{40000} - \frac{\left(s - \frac{r}{2}\right)^2}{10000} = \frac{1 - s^2 + rs}{10000}.$$

This function has a unique maximum at $s = r/2$ and no local maxima.

B2. De Jong's F8 (Modified)

De Jong's F8 is simplified to one variable and converted to a maximization form having all positive values. The original function is

$$f(x_1, \ldots, x_k) = 1 + \sum_{i=1}^{k} \frac{x_i^2}{4000} - \prod_{i=1}^{k} \cos\left(x_i / \sqrt{i}\right) \qquad -512 \le x_i \le 511.$$

Performing the various steps yields

$$f(s) = 1 - \frac{s^2 - rx}{4000} + \cos\left(s - r/2\right).$$

This fitness function has a unique maximum at $s = r/2$ and many local maxima.

B3. Trap Function

Our third function is a trap function [7]. These are defined for binary problems and are given by

$$f(s) = 1 + \begin{cases} \frac{a}{z}(z - m(s)) & m(s) \le z \\ \frac{b}{\ell - z}(m(s) - z) & \text{otherwise} \end{cases}$$

where a and b are positive and z is chosen under various criteria. We use $z = \ell - 1$, a = 10 and b = 12. These choices make f(s) fully deceptive [7].

B4. Mason's Higher Cardinality Deceptive Function

Mason [11] presented a method for constructing higher cardinality deceptive functions of any order. One chooses the maximal point and order of deception desired and then works through an algorithm constructing fitnesses of successively higher order schemata. We constructed this function for $c = 4$, $\ell = 2$ and $i^* = 13$. This same function was used over the binary strings having the same integer value.

Function		Optimal String	r	Cardinalities
De Jong F1	$f(x) = \frac{1 - s^2 + rs}{10000}$	$r/2$	16	2,4
De Jong F8	$f(s) = 1 - \frac{s^2 - rs}{4000} + \cos\left(s - r/2\right)$	$r/2$	16	2,4
Trap Function	$f(s) = 1 + \begin{cases} \frac{a}{z}(z - m(s)) & m(s) \le z \\ \frac{b}{\ell - z}(m(s) - z) & \text{otherwise} \end{cases}$	r-1	16	2,4
Mason Function	$c = 4$ $\ell = 2$ $i^* = 13$	13	16	2,4

Table 3: Test Functions

C. Experiments

For each test function, we explore two natural cardinalities. For each cardinality we ran 2 experiments by having the population size vary from 1 to 2. For each of these we obtained two expected times - one with the initial population generated randomly without replacement and one with the initial population generated randomly with replacement. For each data point, we performed the following enumerative search to determine the best expected waiting time:

Vary χ from 0.01 to 1.0 by step sizes of 0.01 and vary μ from 0.01 to $1/c$ in step sizes of 0.01 and then compute the expected waiting time to the optimal string to find the minimum expected waiting time.

5. Results

Tables 4, 5, 6 and 7 contain the best expected waiting times from each of our runs. The blocks marked by "with (w/o)" mean expected waiting times where the initial population is generated with (with out) replacement.

n	$\ell = 4, c = 2$	$\ell = 2, c = 4$
1	with 15.00	with 34.20
	w/o 15.00	w/o 34.20
2	with 7.25806	with 14.9161
	w/o 7.22581	w/o 14.8313

Table 4: Expected Waiting Times for De Jong's F1

n	$\ell = 4, c = 2$	$\ell = 2, c = 4$
1	with 15.00	with 34.20
	w/o 15.00	w/o 34.20
2	with 7.25806	with 21.9628
	w/o 7.22581	w/o 21.8864

Table 5: Expected Waiting Times for De Jong's F8

n	$\ell = 4, c = 2$	$\ell = 2, c = 4$
1	with 15.00	with 34.20
	w/o 15.00	w/o 34.20
2	with 7.25806	with 10.8265
	w/o 7.22581	w/o 10.7381

Table 6: Expected Waiting Times for the Trap Function

n	$\ell = 4, c = 2$	$\ell = 2, c = 4$
1	with 15.00	with 34.20
	w/o 15.00	w/o 34.20
2	with 7.25806	with 35.1026
	w/o 7.22581	w/o 35.0329

Table 7: Expected Waiting Times for the Mason Function

6. Discussion of Results

A. Fitness Invariance

Notice in Tables 4-7 that the expected waiting times for populations of size one are independent of the fitness function, cardinality and string size. The explanation is as follows. When $n = 1$

$$q_{i,g} = \mathcal{M}\left(\frac{F\phi_i}{e'F\phi_i}\right)_g = \mathcal{M}\left(e_i\right)_g$$

and no fitness information remains.

Also, expected waiting times for binary cardinality problems with populations of size 2 seem invariant with the fitness function. A partial explanation is as follows. In all cases, the minimum expected waiting time was found at $\chi = 1$ and $\mu = 0.5$. At these values

$$M = \frac{1}{2^{\ell}} ee'$$

and

$$q_{i,g} = \mathcal{M}\left(\frac{F\phi_i}{e'F\phi_i}\right)_g = \frac{1}{2^{\ell}} \frac{\phi_i' F}{e'F\phi_i} \sigma_g ee' \sigma_g \frac{F\phi_i}{e'F\phi_i} = \frac{1}{2^{\ell}}.$$

So here also all fitness information is removed. Why the minimum expected waiting time was found at $\chi = 1$ and $\mu = 0.5$ remains unknown.

B. Hypotheses

Expected waiting times all increased with increased cardinality so we cannot reject Hypothesis 1, the principle of minimal alphabets. The same conclusion holds for the stronger statement in Hypothesis 2.

In all cases, the expected waiting times for sampling of the initial population without replacement were less than or equal to that sampled with replacement so we cannot reject Hypothesis 3. Sampling without replacement gives a greater initial diversity which, at least in these small population sizes, appears to give a slight edge to the search.

In all cases, the expected waiting time to see the optimal solution decreased with increased population size, so we cannot reject Hypothesis 4.

In sum, all our hypotheses seem supported by the experiments.

C. Other Observations

Two additional points were observed in our runs. The first was that the results for population size one were invariant with respects to the crossover rate, as one might suspect. Second, the expected waiting times were decreasing in increasing crossover and mutation rates. This latter result was not expected.

7. Summary and Future Directions

In this paper we computed the expected waiting time to see an optimal string for four different test functions under varying cardinalities and population sizes. In all cases, our original hypotheses were supported. In particular, the observed expected waiting times increased with cardinality (consistent with the principle of minimal alphabets) and decreased with population size. Furthermore, the expected waiting times were lower when the initial population was sampled without replacement. Finally, the expected waiting times were all decreasing in increasing crossover and mutation rates.

This study can be extended in many ways. First, larger population sizes, larger string sizes and more families of natural cardinalities should be studied. All of these extensions yield heavy computational demands which need to be solved. One possible solution is to use iterative procedures (see [9] for a start on this problem) to compute the expected waiting times. Also, knowing the optimal mutation and crossover rates can reduce the overall computational effort dramatically. Even using some sort of gradient search instead of an enumeration may yield reasonable computational demands.

Another area that could be explored would be how to handle problems not naturally represented by an alphabet. We outlined two procedures to deal with this issue but haven't explored their impact on expected waiting times.

All of our test problems had a unique optimal solution. Another extension would be to study problems with multiple optima strings.

References

[1] Antonisse, J., "A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint," **Proceedings of the Third International Conference on Genetic Algorithms**, Ed. J. D. Schaffer, Morgan Kaufmann, San Francisco, 1989, pp. 86-97.

[2] Bhattacharyya, S. and G. J. Koehler, "An Analysis of Genetic Algorithms of Cardinality 2^V," **Complex Systems**, 8, 1994, pp. 227-256.

[3] Davis, L., Handbook of Genetic Algorithms, 1991, Van Nostrand Reinhold, New York.

[4] Davis, T., "Toward an Extrapolation of the Simulated Annealing Convergence Theory onto the Simple Genetic Algorithm," 1991, Dissertation, University of Florida.

[5] De Jong, K. An Analysis of the Behavior of a Class of Genetic Adaptive Systems, PhD Thesis, 1975, University of Michigan, Department of Computer and Communication Sciences, Ann Arbor, MI.

[6] De Jong, K. A., W. M. Spears and D. F. Gordon, "Using Markov Chains to Analyze GAFOs," in Foundations of Genetic Algorithms 3, 1995, Morgan Kaufmann, San Francisco, pp. 115-137.

[7] Deb, K. and D. E. Goldberg, "Analyzing Deception in Trap Functions," Foundations of Genetic Algorithms 2, Ed. L. Darrell Whitley, 1993, Morgan Kaufmann, San Francisco, pp. 93-108.

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

[9] Koehler, G. J., "Computing Simple GA Expected Waiting Times,", in review, January, 1997, College of Business, BUS 351, University of Florida, Gainesville, FL 32611.

[10] Koehler, G. J., S. Bhattacharyya, and M. Vose, "General Cardinality Genetic Algorithms," in review, No. 95-06-101, College of Business, BUS 351, University of Florida, Gainesville, FL 32611.

[11] Mason, A. J., "Partition Coefficients, Static Deception and Deceptive Problems for Non-Binary Alphabets," **Proceedings of the Fourth International Conference**

on Genetic Algorithms, Ed. R. K. Belew and L. B. Booker, Morgan Kaufmann, San Francisco, 1991, pp. 210-214.

[12] Nix, A. and M. D. Vose, Modeling Genetic Algorithms with Markov Chains," **Annals of Mathematics and Artificial Intelligence**, 5, 1992, pp. 79-88.

[13] Shaffer, J. D., "Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms," Dissertation, 1984, CS Department, Vanderbilt University, Nashville, TN.

[14] Spears, W. M. and K. A. De Jong,, "Analyzing GAs using Markov Models with Semantically Ordered and Lumped States," forthcoming Foundations of Genetic Algorithms 4, 1996.

[15] Vose, M. D., "Formalizing Genetic Algorithms", **Proceedings of the IEEE Workshop on Genetic Algorithms, Neural Networks, and Simulated Annealing Applied to Signal and Image Processing**, Glasgow, Scotland, May, 1990.

[16] Vose, M. D., The Simple Genetic Algorithm: Foundations and Theory, forthcoming, MIT Press, Cambridge, MA.

[17] Vose, M. D. and G. E. Liepins, "Punctuated Equilibria in Genetic Search**,"** **Complex Systems**, 5, 1991, pp. 31-44.