Syntactic Analysis of Convergence in Genetic Algorithms

Sushil J. Louis
Department of Computer Science
Indiana University
Bloomington, IN 47405
louis@cs.indiana.edu

Gregory J. E. Rawlins
Department of Computer Science
Indiana University
Bloomington, IN 47405
rawlins@cs.indiana.edu

Abstract

We use the average hamming distance of a population as a syntactic metric to obtain probabilistic bounds on the time convergence of genetic algorithms. Analysis of a *flat* function provides worst case time complexity for static functions and gives a theoretical basis to the problem of premature convergence. We suggest simple changes that mitigate this problem and help fight deception. Further, employing linearly computable syntactic information, we can provide upper limits on the time beyond which progress is unlikely on an arbitrary function. Preliminary results support our analysis.

1 INTRODUCTION

A Genetic Algorithm (GA) is a randomized parallel search method modeled on natural selection (Holland, 1975). GAs are being applied to a variety of problems and are becoming an important tool in machine learning and function optimization (Goldberg, 1989). Their beauty lies in their ability to model the robustness, flexibility and graceful degradation of biological systems.

Natural selection uses diversity in a population to produce adaptation. Ignoring the effects of mutation for the present, if there is no diversity there is nothing for natural selection to work on. Since GAs mirror natural selection, we apply the same principles and use a measure of diversity for estimating time to stagnation or convergence. A GA converges when most of the population is identical, or in other words, the diversity is minimal. Using the average hamming distance (hamming average) sampled between all members of a population as a measure of population diversity we derive an upper bound for the time to minimal diversity, when the genetic algorithm may be expected to make no further progress (hamming convergence).

Previous work in GA convergence by Ankenbrandt, and Goldberg and Deb focuses on the time to convergence to a particular allele using fitness ratios to obtain bounds on time complexity (Ankenbrandt 1991; Goldberg and Deb 1991). Since GAs use syntactic information to guide their search, it seems natural to use syntactic metrics in their analysis. Such analysis, using hamming averages, can predict the time beyond which qualitative improvements in the solution are unlikely, however it cannot predict the quality of the converged solution. Our analysis gives a theoretical basis for the popular notion among GA practitioners that selection is overly exploitative. We suggest some remedies based on this analysis.

The next section defines our model of a genetic algorithm and identifies the effect of genetic operators on our diversity metric, which is the hamming average. Subsequently we derive an upper bound on the expected time to convergence. This suggests syntactic remedies to the problem of premature convergence and, as a side effect, how to mitigate deception in GAs. Since crossover does not affect the hamming average we extrapolate the change in the hamming average sampled during the first few generations to predict the hamming average in later generations. Results presented in section seven on a test suite of functions indicate that surprisingly accurate predictions are possible. The last section covers conclusions and directions for further research.

2 GENETIC ALGORITHMS AND HAMMING DISTANCE

A genetic algorithm works with a population and encodes a problem's parameters in a binary string. The initial population is formed by a randomly generated set of strings. Our model of a genetic algorithm assumes proportional selection, n-point crossover and the usual mutation operator. With the GA operators defined we can analyze their effects on the average hamming distance of a population.

The average hamming distance of a population is the average distance between all pairs of strings in a population of size N. As each member of the population is involved in $N-1$ pairs, the sample size from which we calculate the average is:

$$\frac{N(N-1)}{2}$$

Let the length of the member strings in the population be l. The hamming average of the initial population is well approximated by the normal distribution with mean h_0 where

$$h_0 = \frac{l}{2}$$

and standard deviation s_0 given by

$$s_0 = \frac{\sqrt{l}}{2}$$

Ignoring the effect of mutation, the hamming average of a converged population is zero.[1] Given that the initial hamming average is $l/2$ and the final hamming average is zero, the effects of selection and crossover determine the behavior of a genetic algorithm in the intervening time.

3 CROSSOVER AND AVERAGE HAMMING DISTANCE

Assuming that offspring replace their parents during a crossover, all crossover operators can be partitioned into two groups based on whether or not they change the hamming average. If one parent contributes the same alleles to *both* offspring (as in masked crossover (Louis and Rawlins 1991)) the hamming distance between the children is less than the hamming distance between their parents. This leads to a loss of genetic material, reducing population hamming average and resulting in faster hamming convergence. We do not consider such operators in this paper. The vast majority of traditional operators, like one-point, two-point, ... l-point, uniform and punctuated crossover (De Jong and Spears 1991; Schaffer and Morishima 1987, 1988; Syswerda 1989), do not affect the hamming average of a population. Before proving this we introduce some notation. Let the two parents A and B, and their offspring C and D, be denoted by:

$$\begin{array}{c} A = a_1, a_2, \ldots, a_l \\ B = b_1, b_2, \ldots, b_l \\ \text{and} \\ C = c_1, c_2, \ldots, c_l \\ D = d_1, d_2, \ldots, d_l \end{array}$$

Traditional crossover, realized by a binary mask $M = (m_1, m_2, \ldots, m_l)$ of length l, is defined by

$$A \times B \stackrel{M}{\Longrightarrow} C$$

$$\text{where } c_i \leftarrow \left\{ \begin{array}{ll} a_i & \text{if } m_i = 1 \\ b_i & \text{if } m_i = 0 \end{array} \right\} \forall\, i \mid (1 \le i \le l)$$

Similarly

$$B \times A \stackrel{M}{\Longrightarrow} D$$

$$\text{where } d_i \leftarrow \left\{ \begin{array}{ll} b_i & \text{if } m_i = 1 \\ a_i & \text{if } m_i = 0 \end{array} \right\} \forall\, i \mid (1 \le i \le l)$$

M determines the implemented traditional crossover operator. With this definition, we prove the following lemma.

Lemma 1 *Traditional crossover operators do not change the average hamming distance of a given population.*

Proof: We prove that the hamming average in generation $t+1$ is the same as the hamming average at generation t under the action of crossover alone. Assuming

[1] Mutation increases the hamming average of a converged population by an amount $\epsilon > 0$ depending on the probability of mutation.

a binary alphabet $\{a, b\}$, we can express the population hamming average at generation t as the sum of hamming averages of l loci, where l is the length of the chromosome. Letting $h_{i,t}$ stand for the hamming average of the i^{th} locus we have:

$$h_t = \sum_{i=1}^{l} h_{i,t} \tag{1}$$

The hamming average in the next generation is

$$h_{t+1} = \sum_{i=1}^{l} h_{i,t+1}$$

In the absence of selection and mutation, crossover only changes the order in which we sum the contributions at each locus. That is:

$$h_{i,t} = h_{i,t+1}$$

Therefore

$$h_t = h_{t+1}$$

Q.E.D

Having eliminated crossover from consideration since it causes no change in the hamming average, we look to selection as the force responsible for hamming convergence.

4 SELECTION AND AVERAGE HAMMING DISTANCE

Selection is the domain-dependent part of a genetic algorithm. But, independent of the domain, we would like to prove that selection with probability greater than 1/2 reduces the hamming average in successive generations, and then obtain an upper bound on the time to convergence. Obtaining an upper bound is equivalent to assuming the worst possible search space and estimating the time required for finding a point in this space. For a static function, a space on which an algorithm can do no better than random search satisfies this criterion. The *flat* function defined by

$$f(x_i) = \text{constant}$$

contains no useful information for an algorithm searching for a particular point in the space. Thus no algorithm can do better than random search on this function. However, a simple GA without mutation loses diversity and eventually converges. An expression for the time convergence on such a function gives an upper bound on the time complexity of a genetic algorithm on any static function. The GA's convergence on the flat function is caused by random genetic drift where small random variations in allele distribution cause a GA to drift and eventually converge. To derive an upper bound we start with the time for a particular allele to become fixed due to random genetic drift.

If a population of size N contains a proportion p_i of a binary allele i, then the probability that k copies of allele i are produced in the following generation is given by the binomial probability distribution

$$\binom{N}{k} p_i^k (1-p_i)^{N-k}$$

Using this distribution we have to calculate the probability of a particular frequency of occurrence of allele i in subsequent generations. This is a classical problem in population genetics. Although the exact solution is complex, we can approximate the probability that allele frequency takes value p in generation t. Wright's approximation for intermediate allele frequencies and population sizes, as given in Gayle (Gayle 1990), is sufficient for our purposes. Let $f(p,t)$ stand for the probability that allele frequency takes value p in generation t where $0 < p < 1$ then

$$f(p,t) = \frac{6p_0(1-p_0)}{N} \left(1 - \frac{2}{N}\right)^t$$

This specifies the probability that an allele has *not* converged. Therefore the probability that an allele is fixed (converged) at generation t is

$$\mathcal{P}(t) = 1 - f(p,t)$$

Applying this to a genetic algorithm, assuming a randomly instantiated population at $t = 0$, we have

$$\mathcal{P}(t) = 1 - \frac{6p_0(1-p_0)}{N} \left(1 - \frac{2}{N}\right)^t$$

If we assume that alleles consort independently, which is true for a flat function, then the expression for the probability that all alleles are fixed at generation t for a chromosome of length l alleles, is given by

$$\mathcal{P}(t,l) = \left[1 - \frac{6p_0(1-p_0)}{N} \left(1 - \frac{2}{N}\right)^t\right]^l \tag{2}$$

Equation 2 gives us the time to convergence for a genetic algorithm on a flat function and is therefore an upper bound for any static function. For example, on a 50 bit chromosome and a population size of 30, this gives an upper bound of 92% on the probability of convergence in 50 generations. Experimentally, we get between 92% and 73% convergence starting with an initial hamming average of $50/2 = 25$. Previous work by Goldberg and Segrest on genetic drift gives a more exact albeit more computationally taxing expression for time to convergence due to genetic drift (Goldberg and Segrest 1987). They also include the effect of mutation in their analysis, which once again is directly applicable to our problem. Finally, that GAs converge so quickly due to random drift gives a theoretical basis to the often observed problem of premature convergence. We suggest some methods of mitigating this problem in the next section.

4.1 HANDLING PREMATURE CONVERGENCE

Nature uses large population sizes to "solve" the premature convergence problem. This is expensive, furthermore we need not be restricted to nature's methods but can do some genetic engineering of our own. Mutation seems a likely candidate, and in practice, is the usual way of maintaining diversity. However, although high mutation rates may increase diversity, its random nature raises problems. Mutation is as likely to destroy good schemas as bad ones and therefore elitist selection is needed to preserve the best individuals in a population. This works quite well in practice, but is unstructured and cannot insure that all alleles are always present in the population.

Instead of increasing mutation rates, we pick an individual and add its *bit complement* to the population. This ensures that every allele is present and the population spans the entire encoded space of the problem. We can pick the individual to be complemented in a variety of ways depending on the assumptions we make about the search space. Randomly selecting the individual to be complemented makes the least number of assumptions and may be the best strategy in the absence of other information. We could also select the best or worst individual, or use probabilistic methods in choosing whom to complement. Instead of complementing one individual, we can choose to complement a set of individuals, thus spanning the encoded space in many directions. The most general approach is to maintain the complement of every individual in the population, doubling the population size.

The presence of complementary individuals also makes a GA more resistant to deception. Intuitively, since the optimal schema is the complement of the deceptively optimal schema, it will be repeatedly created with high probability as the GA converges to the deceptive optimum (Goldberg et al. 1989). In our experiments we replaced the minimum fitness individual with either the complement of a randomly picked individual in the current population, or the complement of the current best individual. Let l_i represent the number of bits needed to represent variable i in a deceptive problem, then the functions we used can be described as follows

$$\text{Deceptive}(x_i) = \sum_{i=1}^{n} \left\{ \begin{array}{ll} x_i & \text{if } x_i \neq 0 \\ 2^{l_i+2} & \text{if } x_i = 0 \end{array} \right\}$$

We used *Dec1* and *Dec2* which are 10-bit and 20-bit deceptive problems respectively. Letting superscripts denote the number of bits needed for each variable, Dec1 can be described by

$$\text{Dec1: Deceptive}(x_1^2,\ x_2^8)$$

and Dec2 by

$$\text{Dec2: Deceptive}(x_1^2,\ x_2^8,\ x_3^5,\ x_4^5)$$

Figure 1 compares the average fitness after 100 generations of a classical GA (CGA) and a GA with complements (GAC) on a Dec1. The GAC easily outperforms the classical GA, both in average fitness and the number of times the optimum was found. In most experiments the classical GA fails to find the optimum in 11 runs of the GA with a population size of 30 in a 100 generations. Since the converged hamming average for the CGA is very small, we do not expect it to be ever able to find the global optimum. GAC on the other hand easily finds the optimum for Dec1 within a 100 generations and usually finds the optimum for Dec2 within a few

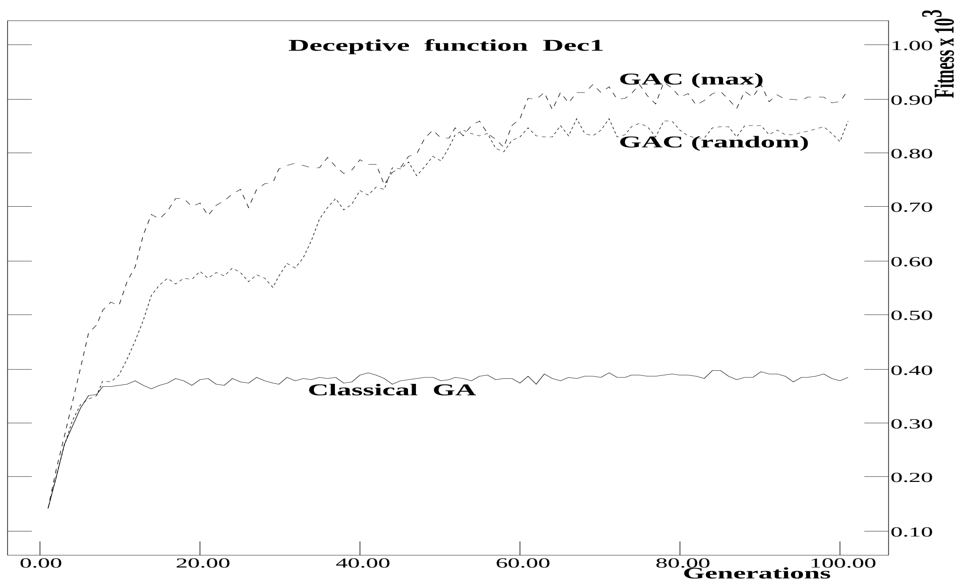

Figure 1: Average fitness over 100 generations of classical GA and GA with complements. GAC (max) replaces the worst individual with the complement of the current best individual. GAC (random) replaces the worst individual with the complement of a random individual in the population

hundred generations. Figure 2 shows the number of times the optimum was found for Dec1 and Dec2 within a total of 100 and 1000 generations respectively.

Although this genetically engineered algorithm can mitigate certain kinds of deception, it is not a panacea and cannot guarantee an optimal solution on all problems.Messy Genetic Algorithms (Goldberg et al. 1989) can make much stronger guarantees but their computational complexity is daunting. Not surprisingly, GAC also does better than a classical GA on Ackley's One Max and no worse on the De Jong test suite (Ackley 1987; De Jong 1975). In problems consisting of mixed deceptive and easy subproblems the GAC still does much better than the classical GA. Finally, GA-hard problems, that are both deceptive and epistatic, may need masked crossover or other length independent recombinant operators to be solved successfully using complements (Louis and Rawlins 1991).

5 PREDICTING TIME TO CONVERGENCE

We now have an upper bound on time to convergence on static functions, but predicting performance on an arbitrary function is more difficult because of the non-linearities introduced by the selection intensity. However, computing the rate of decrease in the hamming average while a GA is working on a particular problem allows us to predict roughly the time to hamming convergence.

We first assume that string similarity implies fitness similarity (the similarity as-

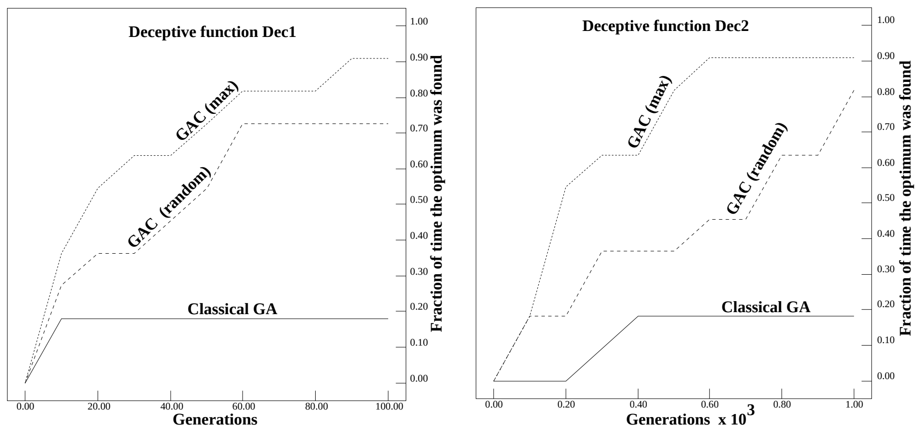

Figure 2: Number of times the optimum was found on **Dec1** and **Dec2** for a classical GA compared with the same statistic for GAs with complements. GAC (max) replaces the worst individual with the complement of the current best individual. GAC (random) replaces the worst individual with the complement of a random individual in the population

sumption). Unimodal functions satisfy this assumption. Even if the function is multimodal, genetic drift, unless countered by niching or other schemes, will ultimately cause behavior that is predictable by the following model. We start with a general equation for the change in hamming average per generation:

$$h_{t+1} = f(h_t)$$

relating h_t, the hamming average in generation t, to the hamming average in the next generation. Two observations guide our choice of the function $f(h_t)$.

1. The schema theorem along with our similarity assumption indicate $f(h_t)$ is linear:
$$h_{t+1} = ah_t + b$$
2. Without mutation the final hamming average is zero, which implies that $b = 0$.

This gives us the equation:

$$h_{t+1} = ah_t$$

We can solve this simple recurrence, and the general solution is given by

$$h_t = \left\{ \begin{array}{ll} l/2 & t = 0 \\ & \\ a^t h_0 & t > 0 \end{array} \right\} \tag{3}$$

We can estimate a by keeping track of the hamming average while running a GA. We want to stop when the hamming average gets under what is computationally feasible to explore with enumeration. Convergence to a hamming average of x with a standard deviation of y where $x + y \approx 10$ will suffice since it is not unreasonable to exhaustively search $2^{10} = 1024$ points in the search space. Mutation however

Table 1: Table comparing actual and predicted hamming averages

Function	*Observed*			*Predicted*	
	h_0	h_{50}	std. dev.	h_{50} $O(N^2)$	h_{50} $O(N)$
Flat	25	4.6	2.4	4.1	4.7
F1	15	1.9	1.2	1.1	3.2
F2	12	2.5	1.1	3.2	7.7
F3	25	3.2	1.9	5.5	3.7
F4	120	20.4	7.6	14.8	17.2
F5	17	2.7	1.9	3.1	2.4
One Max	25	3.7	2.3	2.3	2.1
One Max	100	15.7	8.1	8.5	17.5

may cause the hamming convergence value to be very large (> 10). The value of the hamming average at convergence will then depend on the chromosome length and the probability of mutation.

6 RESULTS

We present results comparing our analytical predictions with empirical performance on the following problems:

1. Flat: $f(x_1, x_2) = 10$, with a 50 bit chromosome.
2. DeJong's five functions: $F1 \ldots F5$
3. One Max : $f(X) = |X|$ where $|X|$ stands for the norm of the bit string X, for both a 50 and a 200 bit chromosome.

The GA population size in all experiments was 30, using roulette wheel selection and two-point crossover with no mutation. Estimating the value for a in equation 3 from the first 10 generations of a run, we predict the hamming average at generation 50. The fourth column uses a computed over all $N(N-1)/2$ pairs and the last column computes a using $\lceil\sqrt{N}\rceil$ strings in the population. The results are summarized in table 1. The experimental values are averages over 10 runs.

The predicted results are very close to the actual values even for this rough approximation. All but two predictions are within one standard deviation of observed value. When this is not the case, the predicted values are greater than the observed ones. This is surprising considering the roughness of our approximation. That good results are obtainable with simplified analysis clearly indicates the validity of our approach.

7 CONCLUSIONS

Analyzing a GA is complicated because of its dependence on the application domain and our lack of understanding of the mapping between our view of the application domain and the GA's view of the same domain. The GA's view of the domain is implicit in the chosen encoding. However, useful analysis can be done on the syntactic information in the encoding, giving us a surprising amount of knowledge. An upper bound on the time complexity followed immediately from considering the effects of drift on allele frequency. The high rate of convergence even on the flat function indicates that selection is overly exploitative. This led us to suggest maintaining complements of individuals in the population thus preserving diversity. Preliminary results indicate that not only do complements maintain diversity but they also mitigate the effects of GA-deception. Fitness-based recombination operators which are more susceptible to deception and premature convergence, but which are not bound to short building blocks for progress, can use complements to maintain diversity and become less susceptible to deception.

Results show that good predictions of time complexity are possible, even from a rough model that uses easily computable syntactic information. The implications of the results on other GA parameters such as population size and mutation rate, both affecting the rate of hamming convergence, are being investigated. Bounds on population size, derived from an analysis of the variance in hamming convergence with variance in population size, can be compared with bounds estimated by other methods.

Refining our model includes using more terms in the hamming average equation. Mutation's effect cannot be underestimated and should be incorporated into our model. The rate of mutation determines the amount of diversity and markedly affects hamming convergence in later generations.

Finally, although we have a bound on the time complexity, qualitative predictions may not be possible with purely syntactic information. The schema theorem links schema proportions to fitness and may thus give us a handle on qualitative predictions.

References

Ackley, D. A., *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, 1987.

Ankenbrandt, C. A., "An Extension to the Theory of Convergence and a Proof of the Time Complexity of Genetic Algorithms," *Foundations of Genetic Algorithms.* Rawlins, Gregory J. E., Editor, Morgan Kauffman, 1991, 53-68.

De Jong, K. A., "An Analysis of the Behavior of a class of Genetic Adaptive Systems," Doctoral Dissertation, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

De Jong, K. A., and Spears, W. M., "An Analysis of Multi-Point Crossover," *Foundations of Genetic Algorithms.* Rawlins, Gregory, J. E., Editor, Morgan Kauffman, 1991, 301-315.

Freund, John. E., *Statistics: A First Course.* Prentice-Hall, 1981.

Gayle, J. S., *Theoretical Population Genetics.* Unwin Hyman, 1990, 56-99.

Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

Goldberg, David E., Korb, Bradley, and Deb, Kalyanmoy. "Messy Genetic Algorithms: Motivation, Analysis, and First Results", TCGA Report No. 89002, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms, 1989.

Goldberg, D. E., and Deb, Kalyanmoy., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Foundations of Genetic Algorithms.* Rawlins, Gregory J. E., Editor, Morgan Kauffman, 1991, 69-93.

Goldberg, D. E., and Segrest, Philip., " Finite Markov Chain Analysis of Genetic Algorithms," *Proceedings of the Second International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates, 1987, 1-8.

Holland, John H., *Adaptation In Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press. 1975.

Louis, Sushil J., and Rawlins, Gregory J. E. "Designer Genetic Algorithms: Genetic Algorithms in Structures Design," *Proceedings of the Fourth International Conference on Genetic Algorithms,* Morgan Kauffman, 1991, 53-60.

Schaffer, David. J., and Morishima, Amy, "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms," *Proceedings of the Second International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates, 1987, 36-40.

Schaffer, J. David, and Morishima, Amy, "Adaptive Knowledge Representation: A Content Sensitive Recombination Mechanism for Genetic Algorithms," *International Journal of Intelligent Systems,* John Wiley & Sons Inc., 1988, Vol 3, 229-246.

Syswerda, G., "Uniform Crossover in Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kauffman, 1989, 2-8.