Cyclic and Chaotic Behavior in Genetic Algorithms

Alden H. Wright
Computer Science Department
University of Montana
Missoula, MT 59812 USA
wright@cs.umt.edu

Alexandru Agapie
Laboratory of Computational Intelligence
Institute for Microtechnologies
Bucharest, PO Box 38-160, 72225 Romania
agapie@imt.pub.ro

Abstract

This paper demonstrates dynamical system models of genetic algorithms that exhibit cycling and chaotic behavior. The genetic algorithm is a binary-representation genetic algorithm with truncation selection and a density-dependent mutation. The density dependent mutation has a separate mutation rate for each bit position which is a function of the level of convergence at that bit position. Density-dependent mutation is a very plausible method to maintain diversity in the genetic algorithm. Further, the introduction of chaos can potentially be used as a source of diversity in a genetic algorithm.

The cycling/chaotic behavior is most easily seen in a 1-bit genetic algorithm, but it also occurs in genetic algorithms over longer strings, and with and without crossover.

Dynamical system models of genetic algorithms model the expected behavior or the algorithm, or the behavior in the limit as the population size goes to infinity. These models are useful because they can show behavior of a genetic algorithm that can be masked by the stochastic effects of running a genetic algorithm with a finite population. The most extensive development of dynamical systems models has been done by Michael Vose and coworkers. (See [Vose and Liepins, 1991], [Vose and Wright, 1994] and [Vose, 1999] for examples.) They have developed an elegant theory of simple genetic algorithms based on *random heuristic search*. Heuristic search theory is based on the idea of a *heuristic map* $\mathcal{G}$, which is a map from a population space to itself. The map $\mathcal{G}$ includes all of the dynamics of the simple genetic algorithm. The map defines a discrete-time dynamical system which we call the *infinite population model*.

The simple genetic algorithm heuristic $\mathcal{G}$ is called *focused* if $\mathcal{G}$ is continuously differentiable and if the sequence $p, \mathcal{G}(p), \mathcal{G}^2(p), \ldots$ converges for every p. In other words, $\mathcal{G}$ is focused if every trajectory of the dynamical system converges to a fixed point.

With one exception, infinite population models of genetic algorithms always seem to converge to a fixed point. The exception is the result of Wright and Bidwell [Wright and Bidwell, 1997], who show stable cycling behavior corresponding to very "weird" mutation and crossover distributions that would never be used in practice. Cycling behavior has also been shown in biological population genetics models [Hastings, 1981].

The random heuristic search model also leads in a natural way to a Markov chain model where the states are (finite) populations. Vose has a number of results that connect the infinite population model to the finite population model in the limit as the population goes to infinity. These theorems assume that the heuristic $\mathcal{G}$ is focused. For example, he shows that ([Vose, 1999], theorem 13.1), the probability of being in a given neighborhood of the set of fixed points can be made arbitrarily high by choosing the population size to be sufficiently large.

Thus, there is a lot of interest in knowing whether the heuristic that defines the infinite population model of a genetic algorithm is focused. This paper gives numerical examples where the infinite population model of a genetic algorithm exhibits stable cycling/chaotic behavior, which implies that the heuristic is not focused.

We expect that the examples of this paper very well could arise in practice if the mutation and selection described in this paper was used. However, they are not examples of the simple genetic algorithm in that a density-dependent mutation scheme is used.

Chaotic behavior could also be useful for restoring diversity in a run of a genetic algorithm that is not making progress. When the GA seems to have converged, the parameters could be adjusted to introduce chaotic behavior, which can move the algorithm from a local optimum.

We are aware of one other paper that discusses chaos (more accurately fractals) and genetic algorithms. Juliany and Vose [Juliany and Vose, 1994] generated fractals by determining the basins of attractions of fixed points of $\mathcal{G}$

1 Notation

Let Ω be the search space of length ℓ binary strings, and let $n = 2^\ell$. For $u, v \in \Omega$, let $u \otimes v$ denote the bitwise-and of u and v, and let $u \oplus v$ denote the bitwise-xor of u and v. Let $\overline{u}$ denote the ones-complement of u, and $\#u$ denote the number of ones in the binary representation of u. Let $\mathbf{1}$ denote the string of ones (or the integer $2^\ell - 1$). Thus, $\overline{u} = \mathbf{1} \oplus u$.

Integers in the interval $[0, n) = [0, 2^\ell)$ are identified with the elements of Ω through their binary representation. If $j \in \Omega$, we assume that j_0 denotes the least significant bit of the binary representation of j and $j_{\ell-1}$ denotes the most significant bit. However, when we write j as a binary string, we will use conventional notation with the least significant bit on the right.

This correspondence allows Ω to be regarded as the product group

$$\Omega = Z_2 \times \ldots \times Z_2$$

where the group operation is $\oplus$. The elements of Ω corresponding to the integers 2^i, $i = 0, \ldots, \ell - 1$ form a natural basis for Ω.

A population for a genetic algorithm over length ℓ binary strings is usually interpreted as a multiset (set with repetitions) of elements of Ω. A population can also be interpreted as a 2^ℓ dimensional incidence vector over the index set Ω: if x is a population vector, where x_i is the proportion of the element $i \in \Omega$ in the population. This implies that $\sum_j x_j = 1$. For example, suppose that $\ell = 2$ so that Ω is identified with the set $\{0, 1, 2, 3\}$. Then the population $\{0, 0, 2, 2, 3\}$ is represented by the population vector p where $p_0 = 2/5$, $p_1 = 0$, $p_2 = 2/5$, and $p_3 = 1/5$. We would also write $p = \langle 2/5, 0, 2/5, 1/5 \rangle^T$.

Let

$$\Lambda = \{p \in R^n : \sum_i p_i = 1 \text{ and } x_i \geq 0 \text{ for all } i \in \Omega\}.$$

Thus any population vector is an element of Λ. Geometrically, Λ can be interpreted as the $n - 1$ dimensional unit simplex in R^n. Note that elements of Λ can be interpreted as probability distributions over Ω.

If $expr$ is a Boolean expression, then

$$[expr] = \begin{cases} 1 & \text{if } expr \text{ is true} \\ 0 & \text{if } expr \text{ is false} \end{cases}$$

If p is a population, $i \in \{0, \ldots, \ell - 1\}$, and $k \in \{0, 1\}$, let

$$S(p, i, k) = \sum_{j \in \Omega} p_j [j_i = k].$$

In other words, $S(p, i, k)$ is the relative frequency of population elements whose ith bit has the value k. For example, if $\ell = 2$ and if $p = \langle 2/5, 0, 2/5, 1/5 \rangle^T$ is the example population described above, then $S(p, 0, 0) = p_0 + p_2 = 4/5$, $S(p, 0, 1) = p_1 + p_3 = 1/5$, $S(p, 1, 0) = p_0 + p_1 = 2/5$, and $S(p, 1, 1) = p_2 + p_3 = 3/5$. $S(p, i, k)$ can be also interpreted as the "schema average" of the schema with one fixed position in position i where the value of that fixed position is k.

In the *random heuristic search* model, a population-based generational search algorithm over Ω is defined by a heuristic function $\mathcal{G} : \Lambda \to \Lambda$. Given a population of size r which is represented by $p \in \Lambda$, the next generation population is obtained by taking r independent samples from the probability distribution $\mathcal{G}(p)$. When the simple genetic algorithm is modeled by random heuristic search, the heuristic function $\mathcal{G}$ can be represented as the composition of a selection heuristic function $\mathcal{F}$, and a mixing heuristic function $\mathcal{M}$. See [Vose, 1999] for more details. In this paper, we express $\mathcal{G}$ as $\mathcal{G}(p) = \mathcal{F} \circ \mathcal{M}(p)$, rather than the more usual $\mathcal{G}(p) = \mathcal{M} \circ \mathcal{F}(p)$.

2 Density-dependent mutation

One of the major practical difficulties in the practical use of genetic algorithms is "premature convergence". The genetic algorithm population loses diversity before sufficient exploration is done to discover the solutions of interest. A number of techniques have been proposed to prevent or slow down premature convergence. These include crowding [DeJong, 1975], sharing [Goldberg and Richardson, 1987], and partial reinitialization [Eshelman, 1991].

In this section, we propose another method to avoid premature convergence, based on maintaining population diversity. The idea is to use a different mutation rate at each string position, and this rate depends on the convergence of the population at that string position. If a string position is highly converged in a population, then a high mutation rate will be used at that string position in the production of the next generation. The theoretical justification of this method can be found in [Leung et al., 1997]; building on the concept of *degree of population diversity*, the authors show that premature convergence on a chromosome location (that is, the probability for allele loss on that position) decreases with population size and increases with $|m - 1/2|$ where m is the mutation rate. As we choose to keep the population size fixed, the suggestion is straightforward: Use population diversity as a quantitative mea-

sure to prevent premature convergence by adaptively varying mutation probability. This also corresponds to the *complementing schema in case of stagnation* procedure introduced by [Agapie and Dediu, 1996] for solving *deceptive* problems.

When there is a mutation rate at each string position, then the process of mutation of a chromosome is to mutate the loci of the chromosome independently. In other words, the probability of flipping the bit at string position i is the mutation rate m_i at that string position.

Next, we show how this adaptive mutation rate can be expressed in the framework of the infinite population model.

Mutation can be described in terms of a probability distribution over mutation masks. If $j \in \Omega$ is a binary string and $u \in \Omega$ is a mutation mask, then the result of mutating j using u is $j \oplus u$. Since mutation masks are elements of the search space, a probability distribution over mutation masks is an element of Λ. Given such a probability distribution $\mu \in \Lambda$, the corresponding mutation heuristic is defined by

$$\mathcal{U}(p)_k = \sum_{u\in\Omega}\sum_{j\in\Omega} \mu_u p_j [u \oplus j = k] = \sum_{j\in\Omega} \mu_{k\oplus j} p_j.$$

In the case where there is a mutation rate m_i for each string position,

$$\mu_u = \prod_{i=0}^{\ell-1} (1 - u_i - m_i + 2u_i m_i)$$

where u_i denotes the bit value of u at string position i, $i = 0, 1, \ldots, \ell - 1$. For example, if $\ell = 5$ and $u = 01001$, then $\mu_u = (1 - m_0)m_1(1 - m_2)(1 - m_3)m_4$.

Under density-dependent mutation, the string position i mutation rate m_i is a function of the current population. In particular, it is a function of the population bit frequencies $S(p, i, 0)$ and $S(p, i, 1)$ at position i. A one way to define such a function is to define a function $r : [0, 1] \to [0, 1/2]$ with the property that $r(1 - x) = r(x)$, and where r has a minimum at $x = 1/2$ and maxima at $x = 0, 1$. Then we define

$$m_i = r(S(p, i, 0)) = r(S(p, i, 1)).$$

One particular family of such functions is defined by:

$$r_{a,b}(x) = 2^{a-1} b \left| x - \frac{1}{2} \right|^a \tag{1}$$

where $a \geq 1$ and $0 \leq b \leq 1$. Under this definition, $r_{a,b}(0) = r_{a,b}(1) = b/2$ and $r_{a,b}(1/2) = 0$. Thus, m_i is $b/2$ when position i has completely converged in the population, and m_i is zero when the bit values at position i have equal frequency.

Figure 1 shows a graph of $r_{4,1}$.

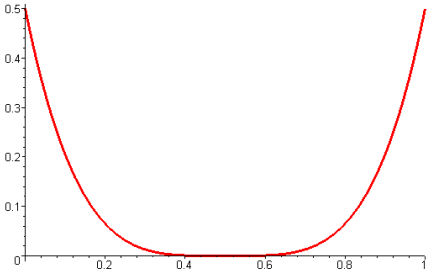

Figure 1: Mutation-adaptation rule: mutation increases whenever diversity is low on a chromosome position.

3 Truncation selection

In this section we show how truncation selection can be modeled by random heuristic search. Denote the population size by r. Under classic truncation selection, a number t with $0 < t < r$ is given. The the current population is ordered by fitness, and the t most fit individuals are selected for reproduction. For simplicity, we assume that all individuals in the population have distinct fitnesses. The most fit t individuals, when represented as a population vector in Λ, define a probability distribution. Under our adaptation of truncation selection to the random heuristic search model, the population at the next step is formed by taking r independent samples from this probability distribution.

This procedure can be defined as a selection heuristic. Without loss of generality, we assume that Ω is ordered so that $f_0 < f_1 < \ldots < f_{n-1}$, where f_j denotes the fitness of $j \in \Omega$. Let $T = t/r$. Then the truncation selection heuristic function $\mathcal{F}_T : \Lambda \to \Lambda$ is defined by:

$$\mathcal{F}_T(p)_k = \tag{2}$$

$$\begin{cases} 0 & \text{if } T < \sum_{k<j} p_j \\ \frac{T - \sum_{k<j} p_j}{T} & \text{if } \sum_{k<j} p_j \leq T < \sum_{k\leq j} p_j \\ \frac{p_k}{T} & \text{if } \sum_{k\leq j} p_j \leq T. \end{cases} \tag{3}$$

It can be verified that this formula agrees with the above procedure for every finite population size. An example will be given in the next section.

4 The 1-bit GA case

We first show how to obtain cycling and chaos in the dynamical system model when the genetic algorithm representation has only a single bit. In this case, the model is 1-dimensional since a population $p = \langle p_0, p_1 \rangle^T$ can be completely described by p_1 since $p_0 = 1 - p_1$. Thus, we identify Λ with $[0, 1]$ under the correspondence $\langle p_0, p_1 \rangle \longleftrightarrow p_1$. Our objective is to describe a heuristic function $\mathcal{G} : [0, 1] \to [0, 1]$ which describes one generation of the GA so that the dynamical system determined by $\mathcal{G}$ exhibits stable cycling.

In the 1-bit case, the truncation selection heuristic defined in the previous section reduces to the following:

$$\mathcal{F}_T(p_1)_1 = \begin{cases} 1 & \text{if } T < p_1 \\ \frac{p_1}{T} & \text{if } p_1 \leq T. \end{cases}$$

In the 1-bit case, the mutation heuristic is:

$$\mathcal{U}(p_0, p_1)_0 = (1-m)p_0 + mp_1$$
$$\mathcal{U}(p_0, p_1)_1 = mp_0 + (1-m)p_1$$

where m is the mutation rate.

Reducing this to the one variable $p = p_1$ gives

$$\mathcal{U}(p) = m - (2m-1)p = m(1-2p) + p$$

If the $r_{a,b}$ function is used for density-dependent mutation, then

$$\begin{aligned} \mathcal{U}_{a,b}(p) &= 2^{a-1}b\left|p - \frac{1}{2}\right|^a (1-2p) + p \\ &= p - \frac{b}{2}|2p-1|^a(2p-1) \end{aligned}$$

It can be verified that $\mathcal{U}_{a,b}$ is continuously differentiable even when $a = 1$ and thus $r_{a,b}$ is not continuously differentiable.

Since there is no crossover possible in the 1-bit case, the heuristic that defines the 1-bit dynamical system describing the genetic algorithm is $\mathcal{G}_{a,b,T} = \mathcal{F}_T \circ \mathcal{U}_{a,b}$.

Figure 2 shows graphs of the selection heuristic $\mathcal{F}_T$ for $T = 7/10$ and the mutation heuristic $\mathcal{U}_{a,b}$ for $a = 2$ and $b = 1$.

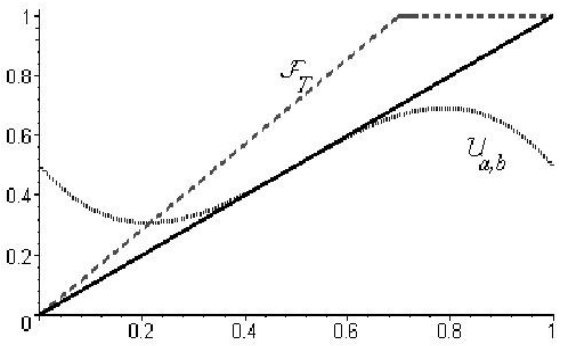

Figure 2: Heuristics $\mathcal{F}_T$ (selection) for $T = 7/10$, res. $\mathcal{U}_{a,b}$ (mutation) for $a = 2$ and $b = 1$

Note that under our assumption that $f_0 < f_1$, if the maximum value of the mutation heuristic function $\mathcal{U}$ is less than T, then the region where the selection heuristic has value 1 is never reached, and the GA heuristic $\mathcal{G}_{a,b,T}$ is continuously differentiable.

We show later that the behavior of $\mathcal{G}$ undergoes a phase transition as $\mathcal{G}$ goes from being continuously differentiable to having a discontinuous derivative. Thus, it is of interest to determine the conditions on a, b, and T that assure that $\mathcal{G}$ is continuously differentiable.

Lemma 1 *If T satisfies the condition*

$$T \geq \frac{1}{2}\left(\frac{1}{b(a+1)}\right)^{\frac{1}{a}}\left(\frac{a}{a+1}\right) + \frac{1}{2}$$

then $\mathcal{G}$ is continuously differentiable.

Proof.

As above, $\mathcal{G}$ is continuously differentiable if and only if the maximum value of $\mathcal{U}$ is less than or equal to T. This maximum value will occur when $p > 1/2$, so we can drop the absolute value from the the formula that defines $\mathcal{U}$. Thus,

$$\mathcal{U}(p) = p - \frac{b}{2}(2p-1)^{a+1}$$

Now we solve the equation $\frac{\partial U}{\partial p} = 0$.

$$\begin{aligned} \frac{\partial U}{\partial p}(p_0) = 0 \quad &\iff \quad (2p_0-1)^a = \frac{1}{b(a+1)} \\ &\iff \quad p_0 = \frac{1}{2}\left(\frac{1}{b(a+1)}\right)^{\frac{1}{a}} + \frac{1}{2} \end{aligned}$$

Then we substitute p_0 into U to obtain the maximum value of $\mathcal{U}$.

$$\begin{aligned} U(p_0) &= \frac{1}{2}\left(\frac{1}{b(a+1)}\right)^{\frac{1}{a}} + \frac{1}{2} - \frac{b}{2}\left(\frac{1}{b(a+1)}\right)^{\frac{a+1}{a}} \\ &= \frac{1}{2}\left(\frac{1}{b(a+1)}\right)^{\frac{1}{a}}\left(\frac{a}{a+1}\right) + \frac{1}{2} \end{aligned}$$

This value gives the minimum for T such that $\mathcal{G}$ is continuously differentiable. □

Figure 3 shows the area of (a, T) space where $\mathcal{G}$ is continuously differentiable.

A fixed point z of a 1-dimensional continuously differentiable discrete-time dynamical system $y \longrightarrow g(y)$ is stable if $|g'(z)| < 1$ and is unstable if $|g'(z)| > 1$. The system is *focused* if f is continuously differentiable and if $\lim_{t\to\infty} g^t(y)$ converges for every starting point y. If $\lim_{t\to\infty} g^t(y) = z$ and if g is continuous, then z is a fixed point of g.

We can now give a more rigorous justification of stable cyclic behavior of the dynamical system defined by $\mathcal{G}$ for some specific values of the parameters.

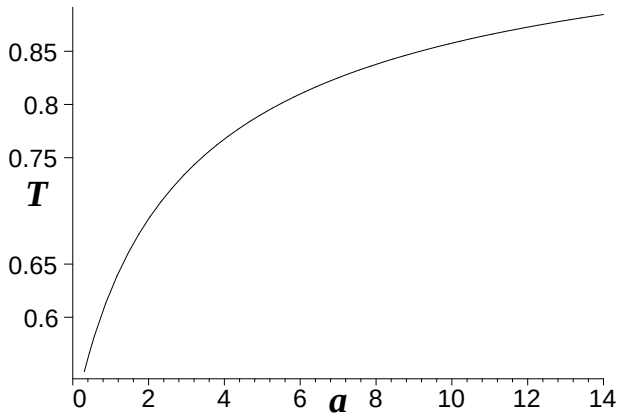

Figure 3: $\mathcal{G}$ is continuously differentiable in the region above the curve.

For the above example with $a = 2$, $b = 1$, and $T = 7/10$, the dynamical system defined by $y \longrightarrow \mathcal{G}(y)$ exhibits stable cycling. $\mathcal{G}$ has a single fixed point at $z = 0.908430$, and $\mathcal{G}'(z) = -1.431114512$, so the fixed point is unstable. $\mathcal{G}^2 = \mathcal{G} \circ \mathcal{G}$ has three fixed points at 0.756838, 0.908430, and 0.984383, and the derivative of $\mathcal{G}^2$ at these points are -0.7721976, 2.0480887, and -0.7721976. Thus, $\mathcal{G}^2$ has two stable fixed points that map to each other under $\mathcal{G}$. This demonstrates that $\mathcal{G}$ exhibits stable cycling of period 2. Figure 4 shows the graphs of the identity function, $\mathcal{G}$, and $\mathcal{G}^2$ for $a = 2$, $b = 1$, and $T = 7/10$.

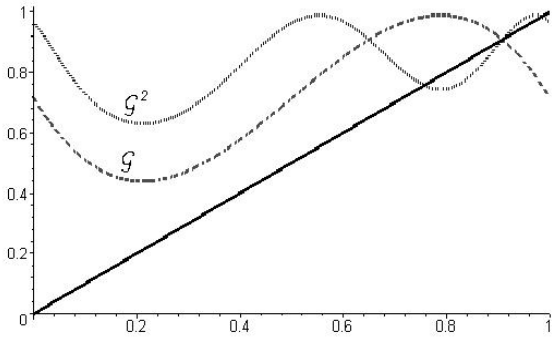

Figure 4: Heuristics $\mathcal{G}$ and $\mathcal{G}^2$ for $a = 2$, $b = 1$, $T = 7/10$

5 Empirical Results Using the Infinite Population Model

We implemented the 1-bit and multi-bit infinite population models in the Maple™programming language. Maple™, along with other mathematical symbolic processing languages, allows for both symbolic processing and arbitrary precision floating point computation.

To assure correctness of the code, we implemented the models in several different ways (1-bit, multi-bit, with and without the Walsh transform), and we cross-checked the results of the different implementations.

5.1 The 1-bit case

For the 1-bit case, we produced what [Peitgen et al., 1992] call *final-state* or *Feignebaum* diagrams. We used the following algorithm for a fixed value of the parameters a, T and b.

1. Choose an initial value p_0 at random from the interval $[0, 1]$.
2. Carry out 100 iterations to compute $p_1, p_2, \ldots, p_{100}$ using $p_{n+1} = \mathcal{G}(p_n)$.
3. Carry out 200 more iterations to compute $p_{101}, \ldots, p_{300}$.
4. Plot $p_{101}, \ldots, p_{300}$ on the diagram.

To produce a diagram, we fixed one of the a and T parameters and varied the other. The b parameter was fixed at 1. Figure 5 shows the diagram with T fixed at $7/8$ and a varying from 5 to 14 with an increment of 0.01. Figure 6 shows the diagram with a fixed at 4 and T varying from 0.6 to 0.99 in increments of 0.001. Both diagrams show period-doubling approach to chaos as the a or T parameter approaches the boundary of the region where $\mathcal{G}$ is continuous. In the region where $\mathcal{G}$ is discontinuous, the behavior seems to be mostly periodic, but with some chaotic deviation from the period behavior for some parameter values.

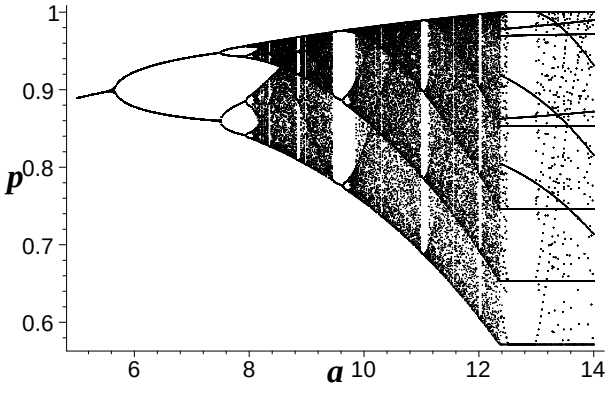

Figure 5: Final-state diagram for $\mathcal{G}$ heuristic for $T = 7/8$, $b = 1$, and values of a from 5 to 14

These diagrams were generated with 100 digits of floating-point precision. However, the diagrams look identical to

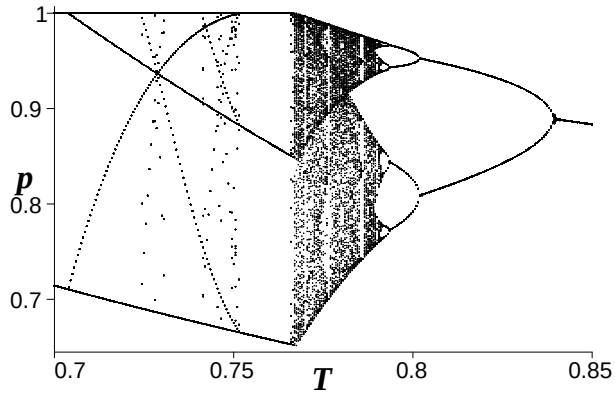

Figure 6: Final-state diagram for $\mathcal{G}$ heuristic for $a = 4$, $b = 1$, and values of T from 0.7 to 0.9

diagrams produced using 10 digits of floating-point precision. Lack of precision means that specific iterates may not be computed correctly (due to the sensitivity of initial conditions due to chaos), but the overall behavior is not affected.

5.2 The multibit case

We also did a number of runs using 2-bit to 4-bit representations, and using one-point and uniform crossover with crossover rates from 0 to 1. The fitness function used assigned one plus the integer value of the binary representation of a string to that string. Thus, the fitness of the 3-bit string 000 was 1, the fitness of 001 was 2, etc.

We found that the behavior of the multi-bit representation models were qualitatively the same (e. g., same period of cycling) as the 1-bit model for the same values of a and T. The presence or absence of crossover and the crossover rate did not affect the results. To be more specific, we ran the model for the combinations of the a and T parameters that are given table 1 for bit lengths of 2 to 4, for crossover rate 1/2, and for both one-point and uniform crossover. To check for cycling of period C, we ran for 100 iterations with a random initial population, and then looked at the 1-norm of the difference between the final population and the population C iterations prior to the final iteration. To check for chaos, we ran both an initial random population and a small random perturbation of this initial random population for 100 iterations, and looked at the 1-norm of the difference between the final populations. The cycling checks were run with 50 decimal digit floating point precision and the chaos checks were run with 100 digit floating point precision.

In the check for cycling, the maximum deviation (1-norm) between the last population and the population C iterations back was 1.3×10^{-6}, and in the check for chaos, a perturbation of size approximately 10^{-50} grew to a difference of at least 10^{-38} after 100 iterations.

$T\backslash a$	4	6	8	10
3/4	10	3	3	3
5/6	2	Chaos	4	4
7/8	1	2	8	Chaos
9/10	1	1	2	4
11/12	1	1	1	2
13/14	1	1	1	1

Table 1: Cycle length or chaotic behavior for 1-bit to 4-bit representations, for different values of the parameters a and T

We did some experiments with the multibit model where the parameters for the bits were set separately. When one bit was set to give cycling behavior and the other bits had a constant mutation rate, the model exhibited cyclic behavior of the same period as when all bits were set to give cycling behavior. And when one bit was set to give chaotic behavior and the remaining bits had constant mutation, the model exhibited chaotic behavior. When bits were set to give cyclic behavior of different periods where the shorter periods were divisors of the longest period, and where $\mathcal{G}$ was continuously differentiable, the longest period resulted. When one bit was set with parameter values that did not make $\mathcal{G}$ continuously differentiable and corresponded to period 3, and other bits were set to give period 10, period 3 resulted.

6 Empirical Results Using A Finite Population GA

We implemented the finite population genetic algorithm that corresponds to the infinite population model described above. To show how the behavior depends on the population size, we did many runs of the case where $r = 5/6$ and $a = 4$. As shown in table 1, the infinite population model has cyclic behavior of period 2 in this case. As a test for cyclic behavior, we looked at when the population average fitness exhibited period 2 cyclic behavior. Let f_t denote the average population fitness at time t. We would say that the GA has cyclic behavior at time t (over 4 generations) if either $f_{t-3} \leq f_{t-2}$, $f_{t-2} \geq f_{t-1}$, and $f_{t-1} \leq f_t$; or if $f_{t-3} \geq f_{t-2}$, $f_{t-2} \leq f_{t-1}$, and $f_{t-1} \geq f_t$. Note that the probability of this happening for a given value of t for a sequence of uniformly distributed random numbers is $1/4$.

Table 6 shows the results of running the GA for 100 runs for 1020 generations with population sizes 100, 250, 1000, and 5000. The fitness function was the same as used for the infinite population model. The string length was 50, and 1-point crossover with a crossover rate of 1/2 was used.

Truncation selection was used with $r = 5/6$, and bitwise density dependent mutation with $a = 4$ was used. The table shows how many of the last 1000 generations exhibited cyclic behavior as defined above.

These results show that the infinite population model makes predictions about a finite population genetic algorithm that can be verified with a population size of 100.

Population size	100	250	1000	5000
# generations	660.8	804.4	968.0	1000.0
Standard error	18.1	16.5	8.7	0.0

Table 2: Number of generations exhibiting cyclic behavior out of 1000

7 Conclusion

We have shown that introducing a bitwise density dependent mutatation in conjunction with truncation selection into a bit-representation genetic algorithm can cause the infinite-population model of this genetic algorithm to exhibit cyclic and chaotic behavior. As the mutation parameter increases, the model goes through a period-doubling approach to chaos.

This work is significant for two reasons.

First, it shows that the very important Vose infinite population model can exhibit a qualitatively different kind of behavior, namely chaos, than has been seen before.

Second, it demonstrates a new way to introduce diversity into an evolutionary computation algorithm, namely the cyclic and chaotic behavior shown in this paper. To follow up on this, more work needs to be done on characterizing the conditions under which cyclic and chaotic behavior can occur.

Acknowledgements

This work was done while the second author was visiting the University of Montana, supported by a COBASE grant from the National Research Council, USA.

References

[Agapie and Dediu, 1996] Agapie, A. and Dediu, H. (1996). GA for deceptive problems: Inverting schemata by a statistical approach. In *Proceedings IEEE International Conf. on Evolutionary Computation (ICEC'96)*, pages 336–340, Nagoya, Japan. IEEE.

[DeJong, 1975] DeJong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI.

[Eshelman, 1991] Eshelman, L. (1991). The CHC adaptive search algorithm: how to have safe search while engaging in nontraditional genetic recombination. In Rawlings, G. J. E., editor, *Foundations of genetic algorithms*, pages 265–283, San Mateo. Morgan Kaufmann.

[Goldberg and Richardson, 1987] Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, N. J. Lawrence Erlbaum Associates.

[Hastings, 1981] Hastings, A. (1981). Stable cycing in discrete-time genetic models. *Proc. Nat. Acad. Sci. USA*, 78:7224–7225.

[Juliany and Vose, 1994] Juliany, J. and Vose, M. D. (1994). The genetic algorithm fractal. *Evolutionary Computation*, 2(2):165–180.

[Leung et al., 1997] Leung, Y., Gao, Y., and Xu, Z. B. (1997). Degree of population diversity - a perspective on premature convergence in genetic algorithms and its markov chain analysis. *IEEE Trans. on Neural Networks*, 8:1165–1176.

[Peitgen et al., 1992] Peitgen, H.-O., J:urgens, H., and Saupe, D. (1992). *Chaos and Fractals, New Frontiers of Science*. Springer-Verlag, New York.

[Vose, 1999] Vose, M. D. (1999). *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA.

[Vose and Liepins, 1991] Vose, M. D. and Liepins, G. E. (1991). Punctuated equilibria in genetic search. *Complex Systems*, 5:31–44.

[Vose and Wright, 1994] Vose, M. D. and Wright, A. H. (1994). Simple genetic algorithms with linear fitness. *Evolutionary Computation*, 4(2):347–368.

[Wright and Bidwell, 1997] Wright, A. H. and Bidwell, G. L. (1997). A search for counterexamples to two conjectures on the simple genetic algorithm. In *Foundations of genetic algorithms 4*, pages 73–84, San Mateo. Morgan Kaufmann.