Markov Chain Models of Genetic Algorithms

Alden H. Wright
Computer Science Dept., Univ. of Montana
Missoula, MT 59812
wright@cs.umt.edu, (406) 243-4790

Yong Zhao
Computer Science Dept., Univ. of Montana
Missoula, MT 59812
yzhao@cs.umt.edu, (406) 243-2883

Abstract

Nix and Vose [Nix and Vose, 1992] modeled the simple genetic algorithm as a Markov chain, where the Markov chain states are populations. Vose has extended this model to a "Random Heuristic Search" model of genetic (and other) algorithms where each individual of the next generation is selected from a probability distribution over individuals in the search space. Many genetic algorithms do not fit this framework. The first part of this paper shows how to use Markov chains to model to a wider class of genetic algorithms, including steady state algorithms and algorithms that use a $(\mu + \lambda)$ selection strategy.

Hill-climbing and strict hill-climbing evolutionary algorithms are defined, and an asymptotic convergence rate is shown for a class of strict hill-climbing algorithms. A strict hill-climbing no-mutation genetic algorithm is given that is guaranteed to converge to the optimum individual for separable fitness functions, and an expected time to convergence is proved.

Nix and Vose [Nix and Vose, 1992] model the "Simple Genetic Algorithm", (generational, fixed-length binary string representation, one-point crossover, proportional selection). The model is "exact" in that the model makes no simplifying assumptions for the class of genetic algorithms modeled. Vose ([Vose, 1996], [Vose, 1999]) has extended the model to the more general Random Heuristic Search model which is described in section 3 of this paper.

Suzuki [Suzuki, 1993] shows how to model a class of elitist genetic algorithms with Markov chains, and gives a result on the number of generations needed for convergence based on the probability of mutation to the optimal string. Aytug and Koehler [Aytug and Koehler, 1996] give a upper bound on the number of generations needed for the simple genetic algorithm to converge to the population that consists entirely of the optimal string, again based on the probability of mutation to the optimal string. Aytug, Bhattacharyya, and Koehler [Aytug et al., 1999] have extended this result to general cardinality alphabet GAs.

1 Notation and terminology

Let Ω denote the search space, and let n denote the cardinality of Ω. If a fixed-length binary string represenation is used, then $\Omega = \{0,1\}^\ell$, where ℓ is the string length. Most of the results of this paper apply to an arbitrary finite search space. We will identify the elements of Ω with the integers in the range $[0, n)$.

If *expr* is an expression that may be true or false, then

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

1.1 Populations

Populations are finite multisets (sets with repeated elements) with elements drawn from Ω. We will represent populations as "incidence vectors" indexed by Ω. If X is a population, then X_i is the number of times that $i \in \Omega$ occurs in X. Thus, if X is a population of size r, $\Sigma X_i = r$. Note that $X + Y$ is the population vector that represents the union of the populations X and Y, and $X \leq Y$ is true if and only if X is a subset of Y.

Let $\mathcal{P}_r$ denote the set of populations of size r with elements taken from Ω. Let $\mathcal{P}_k(X)$ denote the set of subpopulations of X of size k.

For the infinite population model, we want a representation of a population which is independent of the

population size. This can be obtained by dividing the population vector by the population size. If $X \in R^n$ is a population of size r denoted by an upper-case letter, we will denote the population-size-independent representation by the corresponding lower case letter. In other words, $x = X/r$.

Let $\Lambda = \{x \in R^n : x_i \geq 0 \text{ and } \Sigma x_i = 1\}$. Then Λ is the set of population-size-independent represntations of populations. (The set of points of Λ that correspond to finite populations is dense in Λ, so one can think of the elements of Λ as corresponding to infinite populations.) Note that Λ is also the set of probability distributions over Ω. Geometrically, Λ is the unit simplex in R^n.

1.2 The multiple hypergeometric distribution

We will make extensive use of the multiple hypergeometric distribution. If X is a population of size r, then the multiple hypergeometric distribution gives the probability $\rho_X(W)$ of choosing a subpopulation W of size k. The formula is:

$$\rho_X(W) = \frac{\prod_{j \in \Omega} \binom{X_j}{W_j}}{\binom{r}{k}}$$

Note that $\sum_{W \in \mathcal{P}_k(X)} \rho_X(W) = 1$. If $\mathcal{B}$ is a set of subpopulations of X, define $\rho_X(\mathcal{B}) = \sum_{W \in \mathcal{B}} \rho_X(W)$.

2 The Vose Infinite Population Model

The Vose infinite population model is an important tool in modeling the behavior of finite-population genetic algorithms. We give only a brief outline. More details can be found in [Vose, 1999], [Vose, 1996], and [Vose and Wright, 1998].

The Vose infinite population model for the simple genetic algorithm is given by a function $\mathcal{G} : \Lambda \to \Lambda$. $\mathcal{G}$ is called a *heuristic* function. $\mathcal{G}$ is the composition of a function $\mathcal{M} : \Lambda \to \Lambda$ that describes mixing (crossover and mutation), and a function $\mathcal{F} : \Lambda \to \Lambda$ that describes selection. $\mathcal{G} = \mathcal{M} \circ \mathcal{F}$ is described for the binary string representation in [Vose, 1996] and [Vose, 1999], for the c-ary string represtation in [Vose and Wright, 1998] and [Vose, 1999], and for the permutation representation in [Vose and Whitley, 1999]. For a string representation, the $\mathcal{M}$ function can describe crossover and mutation based on crossover and mutation masks, including one-point, two-point, and uniform crossover. The $\mathcal{F}$ function can describe proportional, ranking, and tournament selection.

The $\mathcal{G}$ function has three interpretations. First, it is the limiting behavior of one generation of the simple genetic algorithm as the population size goes to infinity. Second, if the current generation population is x, then in each choice of an individual for the next generation population, the probability that individual $i \in \Omega$ is added is $\mathcal{G}(x)_i$. (See the description of the Random Heuristic Search model below for a more precise statement.) Third, $\mathcal{G}(x)$ is the expected next generation population. (This is shown in [Vose, 1999].)

3 The Vose Random Heuristic Search Model

This model shows how to convert a "heuristic function", such as the $\mathcal{G}$ function for the simple genetic algorithm, into a finite population model. More details are given in [Vose, 1998], [Vose and Wright, 1998], and [Vose, 1999].

The RHS model:

1. Choose an initial random population X of size r.
2. Select r independent random samples from the probability distribution $\mathcal{G}(X/r)$, and let these be a new population Y.
3. Replace X by Y.
4. Go to step 2.

When RHS is used to model the simple genetic algorithm, $\mathcal{G}$ is the infinite population model function described in the previous section. Other random algorithms, such as simulated annealing and genetic programming, can be modeled by using an appropriate $\mathcal{G}$ function.

The RHS model assumes a generational genetic algorithm, and assumes that each element of the next generation population is selected independently from the current population. Many commonly used genetic algorithms, such as steady-state algorithms, do not fit this framework. Section 6 gives some examples.

The simple genetic algorithm was modeled as a Markov chain in [Nix and Vose, 1992]. The states of the Markov chain are populations. There are $N = \binom{n+r-1}{r}$ such populations. The transition probabilities for the Markov chain can be obtained by using the multinomial probability distribution. If X and Y correspond to populations to size r, then the transition

probability from X to Y is given by

$$P(X,Y) = r! \prod_{j \in \Omega} \frac{\mathcal{G}(X/r)_j^{Y_j}}{(Y_j)!}$$

This formula can be understood as follows. The sample space is the set of ordered sequences from Ω of length r. A population can be thought of as an equivalence class of ordered sequences where two sequences are equivalent if one can be reordered into the other. The probability of any one sequence that corresponds to population Y is $\prod \mathcal{G}(X/r)_j^{Y_j}$. The number of sequences corresonding to population Y is $r!/\prod Y_j!$.

In the following, we will want a more general version of the RHS formula that computes the probability of choosing a population Y of size k from a population X of size r using heuristic function $\mathcal{H}$. To this end we define the R function as follows:

$$R(\mathcal{H}, X, Y) = k! \prod_{j \in \Omega} \frac{\mathcal{H}(X/r)_j^{Y_j}}{Y_j!} \tag{1}$$

4 An extended GA framework

The purpose of this section is to give a more general genetic algorithm framework, and then to show how it can be modeled with a Markov chain. In section 6, we will show how this framework can be used to model several important genetic algorithms that are used in practice.

Extended GA model:

1. Choose an initial random population X of size r.
2. Let Y be the population obtained by taking k independent random samples from the probability distribution $\mathcal{G}(X/r)$.
3. Let Z be the population obtained by selecting r individuals from X and Y.
4. Replace X by Z.
5. Go to step 2.

The sampling of step 3 may be done with or without replacement. If the sampling is without replacement, then step 3 can be rephrased as deleting k individuals from $X + Y$. Sometimes the deleted elements are restricted to be from X so that all elements of Y survive into the next generation.

4.1 Step 3 done by sampling from $X+Y$ with replacement

When step 3 of the extended GA model is done by sampling from $X+Y$ with replacement, it is easy to use the RHS model to construct the Markov chain transition probabilities.

The transition probabilities for step 2 of the extended GA model are given by $R(\mathcal{G}, X, Y)$. The transition probabilities for step 3 are given by $R(\mathcal{F}, X + Y, Z)$, where $\mathcal{F}$ is the heuristic function that corresponds to the type of selection used in step 3. As remarked before, Vose has shown how to model proprotional selection, ranking selection, and tournament selection by such a function.

The Markov chain transition probabilities are given by

$$P(X,Z) = \sum_{Y \in \mathcal{P}_k} R(\mathcal{G}, X, Y) R(\mathcal{F}, X + Y, Z)$$

4.2 Step 3 done by sampling from X without replacement

Sampling from X without replacement is equivalent to choosing k elements of X to delete.

Sampling without replacement weighted by fitness is more difficult to model than sampling with replacement. While in principle we could write down the formulas for proportional or ranking selection without replacement, the formulas would be very complicated. Thus, we limit ourselves to truncation selection (selection of the $r-k$ best elements), and random selection.

4.2.1 Truncation selection from X

In this case, the worst k elements are deleted from X to make room for the k elements of Y. In other words, the best $r-k$ elements from X are selected to keep. If Y is the population chosen from X by RHS in step 2 of the framework, and if W is the subpopulation of X consisting of the best $r-k$ individuals of X, then $Z = Y + W$. Thus, $Y = Z - W$, and the transition probability for step 2 is $R(\mathcal{G}, X, Z-W)$. However, it is possible that there may not be a unique subpopulation of the best $r-k$ individuals of X. In this case, we want to average over all such subpopulations.

If $W \in \mathcal{P}_j$, let $F(W)$ be the sum of the fitnesses of the elements of W. Let $\mathcal{B}_j(X) = \{W \in \mathcal{P}_j : W \leq X \text{ and } F(W) \text{ is maximal }\}$. In other words, $\mathcal{B}_j(X)$ is the set of size j subpopulations of X of maximal fitness.

The Markov chain transition probabilities are: $P(X,Z) =$

$$\frac{1}{\rho_X(\mathcal{B}_{r-k}(X))} \sum_{W \in \mathcal{B}_{r-k}(X)} [W \leq Z] R(\mathcal{G}, X, Z - W) \rho_X(W)$$

Proposition 1 *For all $X \in \mathcal{P}_r$, $\sum_{Z \in \mathcal{P}_r} P(X, Z) = 1$.*

Proof. $\sum_{Z \in \mathcal{P}_r} P(X, Z) =$

$$\sum_{Z \in \mathcal{P}_r} \frac{1}{\rho_X(\mathcal{B}_{r-k}(X))} \sum_{W \in \mathcal{B}_{r-k}(X)} [W \leq Z] R(\mathcal{G}, X, Z-W) \rho_X(W)$$

$$= \frac{1}{\rho_X(\mathcal{B}_{r-k}(X))} \sum_{W \in \mathcal{B}_{r-k}(X)} \rho_X(W) \sum_{Z \in \mathcal{P}_r} [W \leq Z] R(\mathcal{G}, X, Z-W)$$

$$= \frac{1}{\rho_X(\mathcal{B}_{r-k}(X))} \sum_{W \in \mathcal{B}_{r-k}(X)} \rho_X(W) \sum_{Y \in \mathcal{P}_k} R(\mathcal{G}, X, Y)$$

$$= \frac{1}{\rho_X(\mathcal{B}_{r-k}(X))} \sum_{W \in \mathcal{B}_{r-k}(X)} \rho_X(W) = 1 \ \blacksquare$$

4.2.2 Random deletion from X

Next, let us consider the case where k random elements from X are selected for deletion in step 3. Random selection without replacement is described by the multiple hypergeometric distribution. Let W be an arbitrary subpopulation of X of size $r-k$. The probability that W is chosen to survive to the next generation is $\rho_X(W)$.

Let $\mathcal{P}_{r-k}(X)$ denote the set of subpopulations of X of size $r-k$. The Markov chain transition probabilities are given by

$$P(X, Z) = \sum_{W \in \mathcal{P}_{r-k}(X)} [W \leq Z] R(\mathcal{G}, X, Z - W) \rho_X(W)$$

4.2.3 Truncation selection from $X + Y$

Suppose that the worst k elements of $X + Y$ are selected for deletion in step 3.

The the Markov chain transition probabilities are given by

$$P(X, Z) = \sum_{Y \in \mathcal{P}_k} \frac{[Z \in \mathcal{B}_r(X + Y)] R(\mathcal{G}, X, Y) \rho_{X+Y}(Z)}{\rho_{X+Y}(\mathcal{B}_r(X + Y))}$$

A more efficient way to compute these probabilities is given by the following algorithm, where P now denotes a 2-dimensional array indexed over populations.

for $X, Z \in \mathcal{P}_r$ do $P[X, Z] \leftarrow 0$
for $X \in \mathcal{P}_r$ do
  for $Y \in \mathcal{P}_k$ do
    for $Z \in \mathcal{B}_r(X + Y)$ do
      $P[X, Z] \leftarrow P[X, Z] + \frac{R(\mathcal{G}, X, Y) \rho_{X+Y}(Z)}{\rho_{X+Y}(\mathcal{B}_r(X+Y))}$

A more elegant alternative is to model this GA as a Markov chain whose states are populations of size $r+k$. Then one step of the Markov chain consists of steps 3, 4, 5, and 2 of the Extended GA framework, in that order. The GA starts with a population X of size $r + k$, selects the best r elements to keep, and uses RHS to select k elements to add to the kept elements. The transition probabilities are:

$$P(X, Z) = \frac{1}{\rho_X(\mathcal{B}_r(X))} \sum_{W \in \mathcal{B}_r(X)} R(\mathcal{G}, W, Z-W) \rho_X(W)$$

5 Restricting Possible Populations

It is possible to restrict the GA to a subset of all populations. For example, many practical genetic algorithms include methods to help maintain the diversity of the population. One method of doing this is to require that all populations consist of distinct individuals. This can be implemented by checking to see if each new individual generated is already in the population, and if it is, redo the process by which the individual was generated. This adds nothing to the number of fitness evaluations required, but would require searching the population as each new individual is generated.

The number of no-duplicate populations is $\binom{n}{r}$, which can be considerably less than the number of multiset populations.

This process can be modeled as a Markov chain by computing the transition probabilities using the formulas for multiset populations, and then normalizing the transtion matrix by dividing each row of the transition matrix by its sum.

6 Modeling practical genetic algorithms

We describe how the extended GA framework models some genetic algorithms that are commonly used in practice.

Whitley's Genitor algorithm ([Whitley, 1989]) was the first "steady state" genetic algorithm. Genitor selects two parent individuals by ranking selection and applies mixing to them to produce one offspring, which replaces the worst element of the population. Genitor fits the framework of subsection 4.2.1 with $k = 1$.

Eshelman's CHC algorithm [Eshelman, 1991] is a generational GA where the best individuals are drawn from the combined parent and offspring population to obtain the next generation. This selection strategy is called the $(\mu + \lambda)$ Evolution Strategy. Duplicates

are removed from the population. Then parents for crossover are chosen randomly from this population. CHC does not use mutation. Up to this point, CHC is accurately modeled by the framework of subsection 4.2.3 and section 5.

However, CHC has some additional features that we do not model. CHC uses a special kind of uniform crossover called HUX where exactly half of the differing bits are swapped. In addition, CHC places restrictions on which strings are allowed to mate. Binary strings must be a certain distance apart before they are allowed to cross. Section 5 models a limited form of this restriction. When the population converges sufficiently, a "cataclysmic mutation" is done: all strings undergo heavy mutation, except that the best string is kept intact. This could be modeled by an inhomogeneous Markov chain.

7 Hill-climbing genetic algorithms

A *hill-climbing* evolutionary algorithm never decreases fitness (or an objective function) on each step. We define a *strict hill-climbing* algorithm as one that does not change its state on a step unless fitness is increased.

Populations can be sorted lexicographically by fitness as follows: Assume that populations are ordered by fitness so that each population corresponds to a fitness sequence $< f_0, f_1, \ldots, f_{r-1} >$ so that $f_i \geq f_{i+1}$. Sort populations first on the on the highest fitness individual, then on the next highest fitness individual, etc. If $f =< f_0, f_1, \ldots >$ and $g =< g_0, g_1, \ldots >$ are two fitness sequences, then $f < g$ if for some i, $f_j = g_j$ for $j < i$ and $f_i < g_i$. The above ordering extends the work of [Suzuki, 1993].

Using this ordering for populations, the Markov transition matrix for a strict hill-climbing algoirthm will be lower triangular. The transition matrix for a non-strict hill-climbing algorithm will be block lower triangular, where the blocks correspond to populations of equal fitness.

For a strict hill-climbing algorithm where we can find a uniform upper bound the probability that the algorithm remains in any transient state, we can give an asymptotic rate of convergence result. In applying the following theorem, Q is the submatrix of the transition matrix corresponding to the transient states. In the asymtotic result, N is considered to be a constant. Theorem 2 is proved in the appendix.

Theorem 2 *Let Q be an N by N nonnegative lower triangular matrix. Assume that $Q_{i,i} \leq u < 1$ for all i, and that $\sum_j Q_{i,j} \leq 1$ for each i. Then for each i, j, and for any $\epsilon > 0$, $(Q^n)_{i,j} \in o((u+\epsilon)^n)$.*

8 A Time-to-Convergence for a no-mutation genetic algorithm

In this section, we derive the expected time to convergence for a no-mutation genetic algorithm applied to separable fitness functions.

8.1 Separable fitness functions

We assume a representation using strings of length ℓ. We assume that the cardinality of the alphabet at each string position is c, and identify the alphabet with the integers $0, 1, \ldots, c-1$. Let $\otimes$ and $\oplus$ denote componentwise multiplication and addition modulo c. Let $\mathbf{0}$ denote the string of zeros, and $\mathbf{1}$ denote the string of ones. If s is a binary string, let $\overline{s} = s \oplus \mathbf{1}$.

Intuitively, a fitness function is *separable* if the string positions can be partitioned into fitness blocks, and the fitness of any string is the sum of the fitness contributions of each block. To make this more precise, associate a binary mask string m_i with each fitness block. Then these masks have the properties that $m_i \otimes m_j = \mathbf{0}$ for $i \neq j$, and $m_1 \oplus m_2 \oplus \ldots \oplus m_B = \mathbf{1}$.

We will say that a fitness function F depends on the positions specified by mask m if for any two strings u and v, $u \otimes m = v \otimes m$ implies $F(u) = F(v)$. A fitness function F is separable over fitness blocks specified by masks $m_1, \ldots, m_B$ if $F = F_1 + \ldots + F_B$ where each F_i depends on the positions specified by mask m_i.

A separable fitness function can be converted into a linear fitness function over an alphabet of higher cardinality. If the mask m_i has K one bits, then there are c^K possible values for the corresponding fitness block. (We will call these values *alleles*.) Thus, if there are B fitness blocks each of whose masks have K one bits, the fitness function can be represented as a sum of functions of length B strings over an alphabet of cardinality c^K.

The following relates crossover to separable fitness functions. The crossover operation on fixed-length strings can be defined in terms of crossover masks. If u and v are parent strings, and if m is a binary crossover mask string, then the children of the crossover operation using mask m are $(u \otimes m) \oplus (v \times \overline{m})$ and $(v \otimes m) \oplus (u \times \overline{m})$.

Proposition 3 *If F is separable over fitness blocks specified by masks $m_1, \ldots, m_B$, and if m has the property that for each $i = 1, 2, \ldots, B$, either $m \otimes m_i = m_i$ or $m \otimes m_i = \mathbf{0}$, the sum of the fitnesses of the parents*

of a crossover operation using m is equal to the sum of the fitnesses of the children.

Proof. It is easy to see that the parents contain the same set of fitness block alleles as the children. Since the fitness is the sum of the contributions of these alleles, the sums of the fitnesses of the two sets are equal. ∎

The class of separable fitness functions includes many of the test functions used in the literature. A linear fitness function is separable over the set of fitness blocks consisting of single string positions. The sum of deceptive trap functions used by Goldberg and his coworkers ([D. E. Goldberg and Deb, 1989] is one example) are separable. The Royal Road function R1 of [Forrest and Mitchell, 1993] is separable. The more general Royal Road functions of [Jones, 1994] have fitness blocks and have the property that the fitness of any string is greater than or equal to the sum of the fitness contributions of each fitness block. The results of this section apply to the these Royal Road fitness functions.

Note that the fitness contributions of a separable fitness function may have different scales.

8.2 A no-mutation genetic algorithm

One of the problems with no-mutation genetic algorithms is that they can get stuck. If the population loses some allele at some string position, then there is no way for crossover operations to recover that allele. We use the idea of Culbertson's GIGA algorithm [Culbertson, 1992] to guarantee that no alleles are lost. When we do a crossover operation, we either replace both parents with both children, or we discard both children. Since the children have the same set of alleles as the parents, this guarantees that alleles are never lost.

The algorithm of this section is designed to optimize a separable fitness function over a particular set of fitness blocks. In practice, we usually do not know the separability structure of a fitness function, and the fitness function may be only approximately separable. Thus, one could run many versions of the algorithm, each oriented to different partitioning of the string positions into fitness blocks.

If all of the fitness blocks are the same size, then we want each population to contain exactly one copy of each allele for each fitness block. To generate the initial population, we generate all possible alleles for each fitness block in random order. Thus, if the maximum length of a fitness block is K, the maximum number of alleles of a fitness block is c^K, and this is the population size.

We need to count the number of populations since the Markov chain has one state per population.

Lemma 4 *If there are B fitness blocks each of size K, then there are $(c^K!)^{B-1}$ populations of size c^K which contain exactly one of each allele for each fitness block.*

Proof. Each population contains one individual with each of the alleles of the first fitness block, so we can order the population by these values. Then the orderings for the alleles for the remaining fitness blocks are arbitrary, and there are $(c^K!)^{B-1}$ choices for these orderings. ∎

Now we describe the algorithm:

1. Choose an initial population as described above.
2. Let u be a maximum fitness individual from the population, and let v be a random individual from the population. These are the parents.
3. Randomly choose a mask m from the masks $m_1, m_2, \ldots, m_B$, and use m as a crossover mask to produce two children, namely $(u \otimes m) \oplus (u \times \overline{m})$ and $(v \otimes m) \oplus (v \times \overline{m})$.
4. If the fitness of the higher-fitness child is greater than the fitness of the higher-fitness parent, remove the parents from the population and replace them with the children.
5. go to step 2.

Theorem 5 *For the above algorithm, An upper bound on the expected time to a population containing the global optimum string is $Bc^K(c^K!)^{B-1}$, where B is the number of fitness blocks and K is the size of each fitness block.*

Proof. If the current population does not contain an optimum fitness individual, we can compute a mimimum for the probability that the children replace the parents. The higher fitness parent must have at least one fitness block which is not at its optimal allele. Some individual in the population has the optimal allele for this block. The probability that the crossover mask corresponds to this fitness block and that this individual is chosen to be the other parent is $1/(Br) = 1/(Bc^K)$.

This algorithm is a strict hill-climbing genetic algorithm and can be modeled in that framework. An upper bound for the expected time to remain in any state is the inverse of a lower bound on the probability of remaining in that state. Thus, an upper bound

on the expected time to stay in a state is Br. Since the number of states is the number of populations, an upper bound on the expected time to converge to a population containing the optimum is given by $Br(c^K!)^{B-1} = Bc^K(c^K!)^{B-1}$. ∎

Remark. Theorem 2 applies to this algorithm. The above proof shows that u in theorem 2 is at most $1 - 1/(Br) = 1 - 1/(Bc^K)$.

Theorem 6 *For fitness block i let δ_i denote the difference between the fitness of the optimal allele (or alleles) and the fitness of the allele with the next highest fitness. Let $\delta = \min(\delta_1, \ldots, \delta_B)$. Let D denote the difference between the fitness of the maximum fitness individual and the fitness of the minimum fitness individual. An upper bound on the expected time to a population containing the global optimum string is $Bc^K\lceil D/\delta \rceil$, where B is the number of fitness blocks and K is the size of each fitness block.*

Proof. Let M denote the fitness of a maximum fitness individual in the search space. We partition the set of populations into categories based on the fitness of a maximum fitness individual in the population. Category j, for $j = 0, 1, \ldots, \lceil D/\delta \rceil$, is the set of populations whose maximum fitness individual has fitness f in the range $M - j\delta \leq f < M - (j-1)\delta$. Let S_j be the random variable which denotes the number of steps that the GA spends in states of category j. If the GA is in category j with $j > 0$, the maximum fitness individual has some fitness block which is not at optimum, and there is a crossover operation that will swap the optimal allele into this block of the maximum fitness individual. This operation results in an increase in fitness of at least δ, and has probability of at least $1/(Br) = 1/(Bc^K)$. The corresponding transition moves the GA into another category. Thus, the expected value of S_j, $j > 0$, is at most Bc^K. The expected time to optimum is $\sum_{j>0} S_j \leq Bc^K\lceil D/\delta \rceil$. ∎

For the simple Royal Road function R1 of [Forrest and Mitchell, 1993], $c = 2$, $\ell = 64$, $B = 8$, and $K = 8$. For each fitness block, the allele of the all ones string has fitness 8, and other alleles have fitness 0, so $\delta = 8$, and $D = 64$. The upper bound for the expected value of the number of steps is $8 \cdot 2^8 \cdot 8 = 16384$.

As mentioned above, we might not want to assume that we know the separability structure of the fitness function. Thus, the algorithm might be run for different separability structures (i. e., different sets of fitness block masks). Note that the condition of proposition 3 gives an test that might quickly show that a fitness function is not separable.

The worst-case assumption is that the algorithm is run to completion for each possible separability structure in some candidate list of these structures. Then the bounds given in theorem 5 and theorem 6 should be multiplied by the number of separability structures. There are some different assumptions that could be made concerning possible separability structures. If the blocks are contiguous, are of the same size K, and blocks might wrap around the end of the string, then there are at most K such structures. If blocks are of size K and not necessarily contiguous, then there are $\frac{\ell!}{B!(K!)^B}$ such structures.

9 Conclusion

We have shown how to model several kinds of genetic algorithms using Markov chains. These include Whitley's Genitor algorithm and other steady state genetic algorithms, and most aspects of Eshelman's CHC algorithm. We have defined hill-climbing and strict hill-climbing genetic algorithms, and proven an asymptotic geometric rate of convergence theorem for strict hill-climbing algorithms.

Previous Markov chain analysis of genetic algorithms has used the properties of mutation to prove convergence bounds. Analyzing crossover and mutation together is difficult, and for most genetic algorithms, crossover by itself leads to the complications of multiple absorbing states. We have defined a no-mutation genetic algorithm that is guaranteed to converge for separable fitness functions. Thus, for this algorithm applied to the appropriate separable fitness function, the only absorbing states are populations that contain optimal individuals. We have two upper bound results for the expected time to convergence. To our knowledge, this is the first such Markov chain analysis of a crossover-based genetic algorithm. Further work needs to be done to apply these results to algorithms that are less oriented to a particular class of fitness functions. In addition, the expected time to convergence results can be restated as results giving the time needed to achieve a given probability of convergence to the optimum.

Acknowledgements

The first author thanks Kevin Wright and Michael Vose for conversations about topics relating to this paper. The authors thank Nikolaus Vonessen for the proof of theorem 2.

References

[Aytug et al., 1999] Aytug, H., Bhattacharyya, S., and Koehler, G. J. (1999). A markov chain analysis of a general cardinality genetic algorithm. *European Journal of Operational Research*, to appear.

[Aytug and Koehler, 1996] Aytug, H. and Koehler, G. J. (1996). Stopping criteria for finite length genetic algorithms. *ORSA Journal on Computing*, 8(2):183–191.

[Culbertson, 1992] Culbertson, J. (1992). Genetic invariance: a new paradigm for genetic algorithm design. Technical Report TR92-02, University of Alberta Computer Science Department, Edmonton, Alberta, Canada.

[D. E. Goldberg and Deb, 1989]
D. E. Goldberg, B. K. and Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530.

[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.

[Forrest and Mitchell, 1993] Forrest, S. and Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In Whitley, L. D., editor, *Foundations of genetic algorithms 2*, pages 109–126, San Mateo. Morgan Kaufmann.

[Jones, 1994] Jones, T. (1994). A description of holland's royal road function. *Evolutionary Computation*, 2(4):409–415.

[Nix and Vose, 1992] Nix, A. E. and Vose, M. D. (1992). Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88.

[Suzuki, 1993] Suzuki, J. (1993). A markov chain analysis on a genetic algorithm. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 146–153, San Mateo. Morgan Kaufman.

[Vose and Whitley, 1999] Vose, M. and Whitley, D. (1999). A formal language for permutation recombination operators. In Banzhaf, W. and Reeves, C., editors, *Foundations of genetic algorithms (FOGA-5)*, San Mateo. Morgan Kaufmann.

[Vose, 1996] Vose, M. D. (1996). Modeling simple genetic algorithms. *Evolutionary Computation*, 3(4):453–472.

[Vose, 1998] Vose, M. D. (1998). Random heuristic search, applications to gas and functions of unitation. Technical Report ut-cs-98-402, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301.

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

[Vose and Wright, 1998] Vose, M. D. and Wright, A. H. (1998). The simple genetic algorithm and the walsh transform: Part I, theory. *Evolutionary Computation*, 6(3):253–273.

[Whitley, 1989] Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–123. Morgan Kaufman.

Appendix

The following proof is due to Nikolaus Vonessen of the Mathematics Dept. of the University of Montana.

Proof of theorem 2. Let I denote the identity matrix, and let W be the matrix of subdiagonal entries of Q. In other words, $W_{i,j} = Q_{i,j}$ if $i > j$ and $W_{i,j} = 0$ otherwise. It is not hard to show that the elements of W^k are less than or equal to 1, and W^N is the zero matrix.

Then $Q \leq uI + W$, and $Q^n \leq (uI + W)^n$. Since uI and W commute, we can use the binomial theorem on this power. For $n \geq N$,

$$\begin{aligned} Q^n \leq (uI+W)^n &= \sum_{k=0}^{n} \binom{n}{k} W^k u^{n-k} \\ &= u^{n-N} \sum_{k=0}^{N-1} \binom{n}{k} W^k u^{N-k} \end{aligned}$$

For $k \leq N$, $\binom{n}{k} \leq n^k \leq n^N$. Thus, for $n \geq N$ and each i, j, $(Q^n)_{i,j} \leq u^{n-N} N n^N$. By using repeated applications of l'Hôpital's rule, we can show that $\lim_{n\to\infty} \frac{n^N u^n}{(u+\epsilon)^n} = 0$. Thus, $(Q^n)_{i,j} \in o((u+\epsilon)^n)$. ∎