GENETIC ALGORITHM WITH ELITIST MODEL AND ITS CONVERGENCE

DINABANDHU BHANDARI, C. A. MURTHY
and SANKAR K. PAL

Machine Intelligence Unit, Indian Statistical Institute
203 B. T. Road, Calcutta, 700 035, India
E-mail: sankar@isical.ernet.in

In this article, the genetic algorithm with elitist model (EGA) is modeled as a finite state Markov chain. A state in the Markov chain denotes a population together with a potential string. Proof for the convergence of an EGA to the best chromosome (string), among all possible chromosomes, is provided here. Mutation operation has been found to be essential for convergence. It has been shown that an EGA converges to the global optimal solution with any choice of initial population.

Keywords: Genetic algorithms, Markov chains, Elitist model, transition probability matrix, single point crossover, global optimal string.

1. INTRODUCTION

Genetic algorithms (GAs) are stochastic search methods based on the principles of natural genetic systems.[5,8,10] They perform a multidimensional search in providing an optimal solution for evaluation (fitness) function of an optimization problem. GAs are empirically found to provide global near optimal solutions of various complex optimization problems in the fields of operations research, VLSI design, pattern recognition, image processing etc.[1-3,11,13,15]

While solving an optimization problem using GAs, each solution is coded as a string (called "chromosome") of finite length over a finite alphabet $\mathcal{A}$. Each string or chromosome is considered as an individual. A collection of M (M is finite) such individuals is called a population. GAs start with a randomly generated population of size M. In each iteration, a new population of the same size is generated from the current population using three basic operations on the individuals of the population. The operators are (i) Reproduction/Selection, (ii) Crossover and (iii) Mutation.

The new population obtained after selection, crossover and mutation is then used to generate another population. Note that the number of possible populations is always finite since $\mathcal{A}$ is a finite set and M is finite. Sometimes the knowledge about the best string obtained so far is preserved within the population. Such a model is called a genetic algorithm with an elitist model or EGA. This paper deals with the convergence of the genetic algorithms with an elitist model (EGAs).

No stopping criterion, which ensures the optimality, is available in the literature.[2,5,6,8,10] Usually, GAs run for a fixed number "nogen" of iterations or terminate if no improvement is found for a fixed number of iterations. The termination conditions do not ensure the convergence of the process to the global optimal solution. It is also not known whether the optimal solution can at all be reached.

International Journal of Pattern Recognition and Artificial Intelligence Vol. 10 No. 6 (1996) 731–747

Some theoretical aspects of genetic algorithms are studied in the literature. The results rely mainly on string representation of solutions and on the notion of a schema.[8,10] Vose interpreted GAs as constrained random walks and generalized the concept of schemata.[14] There are other papers also on the analysis of schema distribution and deceptive problems.[2,4,9,12]

Davis and Principe[6] introduced a Markov chain genetic algorithm model and derived some theoretical results related to the convergence of GAs. They have provided the theory for the existence of the stationary distribution for the transition from one population to another when mutation probability is assumed to be a positive (> 0) constant. They have also provided the sufficient condition, analogous to the simulated annealing, on the mutation probability parameter sequence to ensure that the non-stationary algorithm achieves limiting distribution. But the proof for convergence of genetic algorirthms to the global optimal solution has not been provided.

In this paper, we have shown that an EGA (genetic algorithm with an elitist model) can be viewed as a Markov chain. It has been shown mathematically that the EGAs eventually converge to the global optimal solution (in the sense of best among all possible individuals) if mutation is performed with a positive (> 0) probability.

The basic principles of genetic algorithms with an elitist model are provided in the next section. The mathematical modeling of EGAs and their convergences are discussed in Secs. 3 and 4.

2. BASIC PRINCIPLES OF GENETIC ALGORITHMS

In each iteration t, a genetic algorithm starts with a population of potential solutions in the form of chromosomes or strings, $P^{(t)} = \{S_1^{(t)}, S_2^{(t)}, \ldots, S_M^{(t)}\}$. Each string $S_i^{(t)}$ is evaluated to give some measure of its fitness in terms of fitness function fit. A mating pool (a tentative new population) is formed by selecting the potential (more fit) individuals from $P(t)$. Members of this population undergo reproduction by means of crossover and mutation operations to form a new population of solutions for the next iteration.

A genetic algorithm for a particular problem is described here, considering a problem of maximizing a function $f(x)$, $x \in D$ where D is a finite set. The problem here is to find x_{opt} such that

$$f(x_{opt}) \geq f(x);\ \forall x \in D\,.$$

Note that D is a discrete domain since D is finite. It can also be a subset of any finite dimensional space. In general, if δ is the number of parameters to be found using GA and A_i represents the finite set of possible values of the ith parameter then $D = A_1, \times A_2, \ldots, A_g$.

2.1. Chromosomal Representation and Initial Population

A string over a finite set of alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$ is used as a chromosomal representation of the solution x. The length (L) of the strings depends on the

required precision of the solution. Each string S corresponds to a value x in D and is of the form

$$S = (\beta_L \beta_{L-1} \ldots \beta_2 \beta_1): \quad \beta_i \in \mathcal{A}, \ \forall i .$$

For example, if we consider $\mathcal{A} = \{0, 1\}$ then a binary string of length L is a chromosomal or string representation of a solution.

The total number of strings (i.e. the number of different values for the variable x) is a a^L. GAs find the best (optimal) string among these a^L strings. It is also to be noted that the total number of strings (i.e. a^L) varies with L. In other words, a higher value of L corresponds to a bigger size of domain D. Hence, if x belongs to an interval $[w_1, w_2]$ (i.e. the domain is not discrete) then the choice of L inherently provides a measure of precision for the solution.[10] A random sample of size M is drawn from a^L strings/chromosomes to form the initial population.

2.2. Fitness Function

As mentioned earlier, the evaluation function plays the role of environment and it ranks the solutions in terms of their fitnesses. Evaluation or fitness function *fit* for a string S is equivalent to the function f

$$fit(S) = f(x) ,$$

where the string S corresponds to x.

2.3. Genetic Operators

In every iteration, we evaluate each chromosome of the population using *fit*. Then, a mating pool is generated using selection and the chromosomes of the mating pool undergo crossover and mutation. Moreover, it is customary to preserve the knowledge of the best chromosome obtained so far during the process.

Selection: This operation is an artificial version of natural selection, a Darwinian survival of the fittest among string creatures. In this process, individual strings of the current population are copied into a mating pool with respect to the empirical probability distribution based on their fitness function values. In this paper we have considered the following strategy (*Gen-Pool*) for generating the mating pool.

- Calculate the fitness value $fit(S_i)$ for each chromosome S_i $(i = 1, 2, \ldots, M)$.
- Find the total fitness of the population $Fit = \sum_{i=1}^{M} fit(S_i)$.
- Calculate the probability g_i of selection for S_i $(i = 1, 2, \ldots, M)$ $g_i = \frac{fit(S_i)}{Fit}$.
- Calculate cumulative probability G_i for S_i $(i = 1, 2, \ldots, M)$ $G_i = \sum_{j=1}^{i} g_j$.

Now, the selection of M strings for the mating pool is performed in the following way. For $j = 1, 2, \ldots, M$,

- generate a random number rnd_j from [0,1],
- if $rnd_j \leq G_1$, then select S_1; otherwise select S_i $(2 \leq i \leq M)$ if $G_{i-1} < rnd_j \leq G_i$.

In this process some chromosomes would be selected more than once and the chromosomes with low fitness values would generally die off. Several other strategies for the generation of mating pool are available in the literature.[8,10]

Crossover: Crossover operation exchanges information between two potential strings and generates two offsprings for the next population. Let p be the probability that a given pair of chromosomes take part in the crossover operation. Then the crossover operation between two chromosomes

$$\beta = (\beta_L \beta_{L-1} \cdots \ldots \beta_2 \beta_1)$$

and

$$\gamma = (\gamma_L \gamma_{L-1} \cdots \ldots \gamma_2 \gamma_1)$$

is performed in the following way:

Generate randomly an integer position pos from the range of $[1, L-1]$. Then two chromosomes β and γ are replaced by a pair β' and γ' where

$$\beta' = (\beta_L \beta_{L-1} \ldots \beta_{pos} \gamma_{pos+1} \ldots \gamma_2 \gamma_1)$$

and

$$\gamma' = (\gamma_L \gamma_{L-1} \ldots \gamma_{pos} \beta_{pos+1} \ldots \beta_2 \beta_1).$$

Crossover operation on the mating pool of size M (M is even) can be performed several ways. One of the ways is described below:

- Choose two strings (a pair) randomly from M strings. Choose again a pair of strings from the remaining $(M-2)$ strings. Repeat this process until $\frac{M}{2}$ pairs are obtained.
- For each pair of strings, generate a random number rnd from [0,1]. If $rnd \leq p$ perform crossover; otherwise no crossover is performed.

The crossover operation between two strings, as stated above, is performed at one position. This is referred as *single point crossover*.[10] Some other forms of crossover operation are also available in the literature[8,10] like multiple point, shuffle and uniform crossover. However, in this paper, we are going to deal with the single point crossover operation.

Mutation: Mutation is an occasional random alteration of a character position. It is performed on each character with the probability $q(>0)$. Every character β_i, $i = 1, 2, \ldots, L$ in each chromosome (generated after crossover) has an equal chance to undergo mutation. Mutation is performed in the following way: For every character β_i at the ith position in a chromosome,

- Generate a random number rnd from [0, 1].
- If $rnd \leq q$, mutate the character β_i by replacing it with one of the randomly selected members of the set $(\mathcal{A} - \{\beta_i\})$.

Note that, any string can be generated from any given string by mutation operation. This result is stated in the form of the following lemma.

Lemma 1: Probability of generating any string S_1, from a given string S_2 is greater than zero and its value is $(\frac{q}{a-1})^{\nu}(1-q)^{L-\nu}$, where ν $(0 \leq \nu \leq L)$ is the number of places where those two strings have distinct characters.

Proof: Trivial

Note that q, in general, is taken to be less than $\frac{a-1}{a}$ and hence the minimum probability of obtaining any string from any given is $(\frac{q}{a-1})^{L}$; that is, mutation needs to be performed at every character position of the given string.

The knowledge about the best string obtained so far is usually preserved within the population. The best string of the current population is copied into the new population by replacing the worst if the fitness values of all individuals in the new population is less than the previous best. In this paper, genetic algorithms with this strategy i.e., maintaining the best string within the population are referred as *genetic algorithms with elitist model* or *EGA*. The basic steps in an EGA is described below.

EGA:

1. Generate an initial population Q of size M and calculate the fitness value of each string S of Q.
2. Find the best string S_{cur} of Q. If the best strings are not unique, then call anyone of the best string is Q as S_{cur}.
3. Construct the mating pool using Gen-Pool (S_{cur} belongs to Q). Perform cross-over and mutation operations on the strings of the mating pool and obtain a population Q_{tmp}.
4. Compare the fitness value of each string S of Q_{tmp} with S_{cur}. Replace the worst string of Q_{tmp} with S_{cur} if the fitness value of each string of Q_{tmp} is less than $fit(S_{cur})$; otherwise no replacement takes place in Q_{tmp}. Rename Q_{tmp} as Q.
5. Go to Step 2.

Note: Steps 2, 3 and 4 together make an iteration.

It is to be mentioned here that, one can make other strategies to preserve the best string, obtained so far, in the population. For example, the best string is copied into the population in place of the worst offspring irrespective of the fitness function values of all the offspring or it is copied in place of the worst if the fitness function value of the worst offspring is less than that of the previous best.

2.4. Genetic Parameters

The values for the genetic parameters L, M, p and q have to be chosen "properly" before performing those operations. The value of L depends on the required precision of the solution (as stated earlier). It has already been stated that M is taken as an even integer. No guidelines exist in the literature for choosing the "appropriate" value of M. Nevertheless any even integer value for $M(\geq 2)$ will serve the purpose of this article. This paper deals with the single point crossover and the probability

(p) of performing a crossover operation is taken to be any value between 0.0 and 1.0. The mutation probability q is taken to be in the interval $(0, \frac{a-1}{a}]$. It is also assumed in this paper that the crossover and mutation probabilities (p and q) remain fixed during the process.

In an iteration, the genetic operations are performed on a population and they result in a new population. In the next iteration, the new population will be subjected to these operations and the process goes on. A genetic algorithm is said to converge to a global optimal string if the probability of reaching a population containing an optimal string goes to one as the number of iterations goes to infinity.

Note that, if genetic algorithms are, at all, useful for optimization problems then they should converge to an optimal solution. This convergence to an optimal solution should not be dependent upon the choice of the initial population. In order to prove the convergence, we have modeled the process in a genetic algorithm as a Markov chain. The model is similar to that of Davis and Principe.[6]

3. MATHEMATICAL MODELLING OF EGA

Strings of length L are considered over a finite alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$ to represent a solution and hence the number of possible strings is a^L, where $a(= |\mathcal{A}|)$ is the cardinality of $\mathcal{A}$. Let $\mathcal{S}$ be the collection of all such strings S. The number of strings in S is finite and each string has a fitness value in terms of *fit*. So there exists an ordered relation between any two of them. More specifically, given two strings $S_1, S_2 \in \mathcal{S}$ there exists exactly one of the following relations:

$$fit(S_1) > fit(S_2)$$
$$fit(S_1) = fit(S_2)$$
$$fit(S_1) < fit(S_2).$$

3.1. Classification of Strings

Let $\mathcal{C} = \{fit(S) : S \in \mathcal{S}\}$. The number of elements in $\mathcal{C}$ is less than or equal to a a^L, since the cardinality of $\mathcal{S}$ is a^L. Let the number of elements in $\mathcal{C}$ be s ($s \leq a^L$). Then $\mathcal{C}$ can be written as

$$\mathcal{C} = \{F_1, F_2, \ldots, F_s\}, \quad \text{where} \quad F_1 > F_2 > \cdots > F_s.$$

This immediately gives us the opportunity to partition $\mathcal{S}$ into a collection of non-empty subsets $\{\mathcal{S}_i\}$ where

$$\mathcal{S}_i = \{S : S \in \mathcal{S} \quad \text{and} \quad fit(S) = F_i\} \quad \forall i = 1, 2, \ldots, s.$$

Let the number of elements of $\mathcal{S}_i$ be u_i then $\sum_{i=1}^{s} u_i = a^L$. It is clear that

$$\mathcal{S}_i \neq \phi, \quad \forall_i = 1, 2, \ldots, s; \ \mathcal{S}_i \cap \mathcal{S}_j = \phi \quad \text{and} \quad \cup_{i=1}^{s} \mathcal{S}_i = \mathcal{S}.$$

Thus for any two strings $S_1 \in \mathcal{S}_i$ and $S_2 \in \mathcal{S}_j$

$$fit(S_1) > fit(S_2) \quad \text{if } i < j,$$
$$fit(S_1) = fit(S_2) \quad \text{if } i = j$$

and

$$fit(S_1) < fit(S_2) \quad \text{if } i > j.$$

This partition enables us to classify all the strings $S \in \mathcal{S}$ so that $\mathcal{S}_1$ consists of all strings which produce the highest possible fitness value F_1. Similarly, $\mathcal{S}_2$ consists of all strings which produce the second highest fitness value F_2 and so on.

3.2. Partitioning of Populations

A population Q is a collection of M strings, where each string is taken from $\mathcal{S}$. Note that Q may contain multiple copies of one or more strings. In particular, M copies of a single string may constitute a population. In general we shall define a population Q in the following way:

$$Q = \{S_1, S_1, \ldots, (\sigma_1 \text{ times}), S_2, S_2, \ldots (\sigma_2 \text{ times}), \ldots, S_\xi, S_\xi, \ldots (\sigma_\xi \text{ times}) : S_i \in \mathcal{S};$$
$$\sigma_i \geq 1; \text{ for all } i = 1, 2, \ldots, \xi; S_i \neq S_j, \text{ for all } i \neq j; \text{ and } \sum_{i=1}^{\xi} \sigma_i = M\}.$$

Two populations Q_1 and Q_2 are, intuitively, equal if both of them contain the same number of copies of the same string. Thus the definition for equality of two populations can be written as:

Definition 1: Two populations Q_1 and Q_2 are said to be equal if for each $S_1 \in Q_1$ and S_1 appears σ_1 times in $Q_1 \iff S_1 \in Q_2$ and S_1 appears σ_1 times in Q_2.

In other words, if two populations Q_1 and Q_2 are equal then both contain σ_i copies of string S_i for all $i = 1, 2, \ldots, \xi$; $\xi \leq M$ such that $\sum_{j=1}^{\xi} = M$. For example,

$$Q_1 = \{S_1, S_1, S_2, S_2\} \neq Q_2 = \{S_1, S_2, S_2, S_2\}.$$
$$Q_1 = \{S_1, S_1, S_2, S_2\} = \{S_1, S_2, S_1, S_2\} = \{S_2, S_1, S_2, S_1\}.$$

From the above definition, the total number of distinct populations is[6]:

$$\mathcal{N} = \binom{a^L + M - 1}{M}.$$

Let $\mathcal{Q}$ be the collection of all such distinct populations, i.e., if Q is a population then $Q \in \mathcal{Q}$.

Let us now define the fitness function value of a population, denoted by $fit(Q)$ in the following way:

Definition 2: $fit(Q) = \max_{S \in Q} fit(S)$.

Note that, $F_s \leq fit(Q) \leq F_1$, for all $Q \in \mathcal{Q}$. Thus one can consider the collection of populations having the same fitness function value as a set. It may be denoted by E_i, $i = 1, 2, \ldots, s$ and can be formally defined as:

$$E_i = \{Q : Q \in \mathcal{Q} \text{ and } fit(Q) = F_i\} \quad i = 1, 2, \ldots, s.$$

In other words,

$$E_1 = \{Q : Q \in \mathcal{Q} \text{ and there exists at least one string } S \in \mathcal{Q} \text{ such that } fit(S) = F_1\}$$

and

$$E_i = \{Q : Q \in \mathcal{Q} \text{ and there exists at least one string } S \in Q \text{ such that } fit(S) = F_i \text{ and } S \in Q \Longrightarrow S \in (\cup_{j=1}^{i-1} S_j)^c\};\ i = 2, 3, \ldots, s.$$

Proposition 1: $E_i \cap E_j = \phi$, for $i \neq j$ and $\cup_{i=1}^{s} E_i = \mathcal{Q}$.

Proof: Follows from the definition of E_i's.

Note that, from Proposition 1, E_i's define a partition on $\mathcal{Q}$.

Let $e_i (> 0)$ be the number of populations in E_i and hence $\sum_{i=1}^{s} e_i = \mathcal{N}$. Let Q_{ij} be the jth population of E_i; $j = 1, 2, \ldots, e_i$ and $i = 1, 2, \ldots, s$, i.e., in other words, any population in $\mathcal{Q}$ is represented as Q_{ij} for some i and j.

3.3. Transitions Between Two Populations

In any iteration or generation, the genetic operators (selection, crossover and mutation) create a population Q_{kl}; $l = 1, 2, \ldots, e_k$ and $k = 1, 2, \ldots, s$; from some Q_{ij}. Since in the present process we are preserving the previous best in the population, the genetic operators cannot generate a population Q_{kl} from Q_{ij}, if $k > i$. Because the fitness value of the new population is at least $fit(Q_{ij}) = F_i$. The creation of a population Q_{kl} from Q_{ij} can be viewed as a transition from Q_{ij} to Q_{kl}.

Let $p_{ij.kl}$ be the probability that the genetic operators result in the population $Q_{kl} \in E_k$ from $Q_{ij} \in E_i$, $j = 1, 2, \ldots, e_i$; $l = 1, 2, \ldots, e_k$; $i, k = 1, 2, \ldots, s$. Let $P_{ij.k}$ denote the probability of transition from Q_{ij} to any population in E_k. This can be obtained by summing $p_{ij.kl}$ over l, i.e.,

$$p_{ij.k} = \sum_{l=1}^{e_k} p_{ij.kl};\ j = 1, 2, \ldots, e_i;\ i,\ k = 1, 2, \ldots, s.$$

Hence,

$$\sum_{k=1}^{s} p_{ij.k} = 1; \quad \text{for all } j = 1, 2, \ldots, e_i \text{ and } i = 1, 2, \ldots, s;$$

Now we are in a position to state the following theorem.

Theorem 1: For all $j = 1, 2, \ldots, e_i$ and $i = 1, 2, \ldots, s$:

$$\begin{aligned} p_{ij\,k} &> 0 \qquad \text{if } k \leq i \\ &= 0 \qquad \text{otherwise}. \end{aligned}$$

Proof: $Q_{ij} \in E_i \Longleftrightarrow$ there exists at least one string S_0 such that $fit(S_0) = F_i$ and $S \in Q_{ij} \Longrightarrow S \in (\cup_{t=1}^{i-1} \mathcal{S}_t)^c = \cup_{t=i}^{s} \mathcal{S}_t$.

Let $Q = \{S_m : m = 1, 2, \ldots, M\}$ be a population generated using the genetic operators on Q_{ij}. Note that, the best string $S_0 \in Q_{ij}$ $(fit(S_0) = F_i)$ is copied in the generated population Q if the fitness values of all the generated offspring are less than F_i and hence

$$\max_m fit(S_m) \geq F_i\,.$$

This implies $p_{ij.kl} = 0$ for all $l = 1, 2, \ldots, e_k$; if $k > i$, and consequently

$$p_{ij.k} = \sum_{l=1}^{e_k} p_{ij.kl} = 0; \quad \forall k > i\,.$$

Now for $k \leq i$, consider a population, say Q_{kl}, which contains M copies of a string S' such that $fit(S') = F_k$. Since $k \leq i$, $fit(S) \leq fit(S')$ for all $S \in Q_{ij}$. If all the strings $S \in Q_{ij}$ are changed to S' by the genetic operators then we will not copy S_0, the best string of Q_{ij}, in the new population. We will show that the probability of such a transition is greater than 0. It is clear that the probability will be minimum (since $0 < q \leq \frac{a-1}{a}$) if we need to mutate each character of each string present in Q_{ij} after the selection and crossover operations. Since there are ML characters in a population, the minimum probability to obtain Q_{kl} from Q_{ij} is $(\frac{q}{a-1})^{ML}$, i.e., $p_{ij.kl} \geq (\frac{q}{a-1})^{ML}$.

Let $Q_{kl_1} \in E_k$ where $l_1 \neq l$ then $p_{ij.kl_1} \geq (\frac{q}{a-1})^{ML}$ (by using a similar argument). Hence,

$$p_{ij.k} = \sum_{l_1=1}^{e_k} p_{ij.kl_1} \geq \sum_{l_1=1}^{e_k} \min_{l_1} p_{ij.kl_1} \geq e_k \cdot \left(\frac{q}{a-1}\right)^{ML} > 0, \quad \text{for all } k \leq i\,.$$

It follows from the above theorem that there is always a positive (> 0) probability to result in one of the populations Q_{kl} of E_k from $Q_{ij} \in E_i$; if $k \leq i$ and zero probability if $k > i$. Thus, in particular, once GAs enter a population $Q \in E_1$ they will always be in E_1, i.e., they will never go out of E_1.

Figure 1 depicts a clear view of transitions in a genetic algorithm with elitist model (EGA). Here each node E_i; $i = 1, 2, \ldots, s$ represents a collection of states (populations) which contain at least one string S of $\mathcal{S}_i$. The nodes are connected by directed links as shown in the figure. Each connection indicates that there exists a positive (> 0) transition probability, i.e., chance of reaching to a state of a node

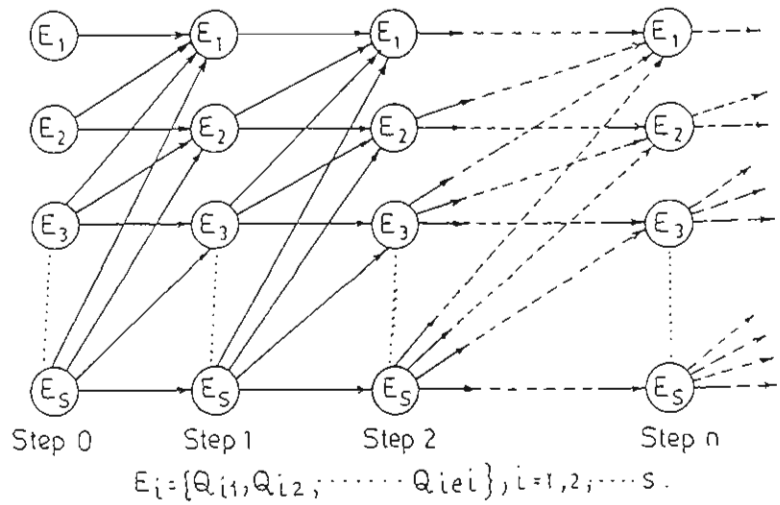

Fig. 1. Transitions between E_i's in EGA.

E_k in iteration $t+1$ from a state $Q_{ij} \in E_i$ in tth iteration. Thus the process of transitions from one population to another can be viewed as a Markov chain.

4. MARKOV CHAIN AND EGA

4.1 EGA as a Markov Chain

It is clear from Sec. 3 that one can consider any population Q_{ij} as a state of a Markov chain. Let η_{ij} be the probability of occurrence of the population Q_{ij}; $j = 1, 2, \ldots, e_i$ and $i = 1, 2, \ldots, s$ at the initial step. Then for a GA,

$$\eta_{ij} \geq 0; \quad \text{for all } j = 1, 2, \ldots, e_i;\ i = 1, 2, \ldots, s;$$

and

$$\sum_{i=1}^{s} \sum_{j=1}^{e_i} \eta_{ij} = 1 . \tag{1}$$

The probabilities $p_{ij\,kl}$, in Sec. 3, represent the transition probabilities from population Q_{ij} to Q_{kl} (or the transition from the state "ij" to the state "kl"). The populations of a GA together with $\{\eta_{ij}\}$ and the conditional probabilities $p_{ij\,kl}$ constitute a Markov chain.[7] In order to denote the transition probability matrix for this Markov chain, let us order the states as $Q_{11}, Q_{12}, \ldots, Q_{1e_1}, Q_{21}, Q_{22}, \ldots, Q_{2e_2}, \ldots, Q_{s1}, Q_{s2}, \ldots, Q_{se_s}$. The number of rows (or columns) in the transition probability matrix $\mathcal{P}$ is $\mathcal{N}$. Every element of $\mathcal{P}$ is $p_{ij\,kl}$, which denotes the probability of transition from Q_{ij} to Q_{kl}. Let $p^{(n)}_{ij,kl}$ be the probability that GA results in Q_{kl} at the nth step given that the initial state is Q_{ij}; then the n-step transition probability matrix $\mathcal{P}^{(n)}$ is $\mathcal{P}^n$.[7] Let

$$p^{(n)}_{ij,k} = \sum_{l} p^{(n)}_{ij,kl} . \tag{2}$$

$p_{ij,k}^{(n)}$ denotes the probability of reaching a population in E_k in n steps with the starting population as Q_{ij}.

In a Markov chain, a set C of states is said to be closed if no state outside C can be reached from any state in C.[7] Note that, no state or population outside E_1 can be obtained using genetic operators from any state $Q \in E_1$ (Theorem 1) and hence the following result:

Result 1: E_1 is closed.

To show the convergence of GAs to a global optimal solution it is sufficient to show that starting from any state Q_{ij}, GAs eventually result in one of the states Q_{1l}; $l = 1, 2, \ldots, e_1$: of E_1. In other words, as n goes to infinity, $p_{ij,kl}^{(n)} \to 0$ $\forall k \geq 2$ and for all i and j.

Theorem 2: For an EGA with the probability of mutation $q \in (0, \frac{a-1}{a}]$,

$$\lim_{n \to \infty} p_{ij,k}^{(n)} = 0 \quad \text{for } 2 \leq k \leq s;\ \forall j = 1, 2, \ldots, e_i \ \text{ and } i = 1, 2, \ldots, s.$$

Hence, $\lim_{n \to \infty} p_{ij,1}^{(n)} = 1 \quad \forall j = 1, 2, \ldots, e_i \ \text{ and } \ i = 1, 2, \ldots, s.$

Proof: It is known that $p_{ij,k}^{(n)} \geq 0$ and $p_{ij,1} > 0$ for all $j = 1, 2, \ldots, e_i$ and $i = 1, 2, \ldots, s$ (from Theorem 1).

Let

$$\max_{i,j}(1 - p_{ij,1}) = \delta. \tag{3}$$

Note that $\delta < 1$ since $\min_{i,j} p_{ij,1} > 0$. Now,

$$\sum_{k \neq 1} p_{ij,k}^{(1)} = \sum_{k=2}^{s} p_{ij,k} = 1 - p_{ij\,1}; \tag{4}$$

$$\begin{aligned}
\sum_{k \neq 1} p_{ij,k}^{(2)} &= \sum_{k=2}^{s} \sum_{i_1 \neq 1} \sum_{j_1 = 1}^{e_{i_1}} p_{ij,i_1 j_1} p_{i_1 j_1,k} \quad (\text{since, } p_{1 j_1,k} = 0 \ \text{ for } \ k > 1) \\
&= \sum_{i_1 \neq 1} \sum_{j_1 = 1}^{e_{i_1}} p_{ij,i_1 j_1} \sum_{k=2}^{s} p_{i_1 j_1,k} \\
&= \sum_{i_1 \neq 1} \sum_{j_1 = 1}^{e_{i_1}} p_{ij,i_1 j_1} (1 - p_{i_1 j_1,1}) \\
&\leq \delta \sum_{i_1 \neq 1} \sum_{j_1 = 1}^{e_{i_1}} p_{ij,i_1 j_1} \\
&= \delta \sum_{i_1 \neq 1} p_{ij,i_1} \quad (\text{from } (3)) \\
&= \delta(1 - p_{ij,1}); \qquad (5)
\end{aligned}$$

similarly,

$$\begin{aligned}
\sum_{k\neq 1} p_{ij\,k}^{(3)} &= \sum_{k=2}^{s}\sum_{i_1\neq 1}\sum_{j_1=1}^{e_{i_1}} p_{ij\,i_1j_1} p_{i_1j_1\,k}^{(2)} \\
&= \sum_{i_1\neq 1}\sum_{j_1=1}^{e_{i_1}} p_{ij,i_1j_1} \sum_{k=2}^{s} p_{i_1j_1,k}^{(2)} \\
&\leq \sum_{i_1\neq 1}\sum_{j_1=1}^{e_{i_1}} p_{ij,i_1j_1}\delta(1-p_{i_1j_1,1}) \text{ (from (5))} \\
&\leq \delta^2 \sum_{i_1\neq 1}\sum_{j_1=1}^{e_{i_1}} p_{ij\,i_1j_1} \\
&= \delta^2 \sum_{i_1\neq 1} p_{ij\,i_1} \\
&= \delta^2(1-p_{ij,1}),
\end{aligned}$$

and in general, by using mathematical induction,

$$\begin{aligned}
\sum_{k\neq 1} p_{ij,k}^{(n+1)} &= \sum_{k=2}^{s}\sum_{i_1\neq 1}\sum_{j_1=1}^{e_{i_1}} p_{ij\,i_1j_1}^{(n)} p_{i_1j_1\,k} \\
&\leq \delta^n(1-p_{ij\,1}). \qquad (6)
\end{aligned}$$

Note that $\delta^n(1-p_{ij,1}) \to 0$ as $n \to \infty$ since $0 \leq \delta < 1$. Hence $\sum_{k\neq 1} p_{ij\,k}^{(n+1)} \to 0$ as $n \to \infty$. Which, immediately implies $\lim_{n\to\infty} p_{ij\,k}^{(n)} = 0$ for $2 \leq k \leq s$ for all i and j. It is clear that,

$$\begin{aligned}
\lim_{n\to\infty} p_{ij,1}^{(n)} &= \lim_{n\to\infty}\left(1-\sum_{k\neq 1} p_{ij\,k}^{(n)}\right) \\
&= 1
\end{aligned}$$

Following conclusions can be made from Theorem 1 (Sec. 3) and Theorem 2.

- From $Q_{1j} \in E_1$, one can at most reach to $Q_{1j_1} \in E_1$, using genetic operators; since $p_{ij\,kl} = 0 \quad \forall k > 1$ and $l = 1, 2, \ldots, e_i$.
- For any state (or population) Q_{ij}; $i \geq 2$, $p_{ij\,k}^{(n)} \to 0$ as $n \to \infty$, $\forall k \geq 2$. In other words, for a sufficiently large number of iterations Q_{ij} will result in Q_{1l} for some $l = 1, 2, \ldots, e_1$, i.e., the convergence to optimal string is assured with any initial population.
- The proof is independent of the crossover operation; but mutation should be performed with probability $q > 0$. Moreover, in each iteration, the knowledge of the best string obtained in the previous iteration is preserved within the population.

- Transition probabilities from one population to another depend on the process of generation of the mating pool. Nevertheless, these probabilities will always obey Theorem 1 due to the mutation operation.
- The structures of E_is depend on the characteristics of the fitness function. Hence the transition probabilities vary for different fitness functions even if the string length and the set of alphabet $\mathcal{A}$ is kept constant.
- $Q_{i,j}$ will eventually result in a population containing the best string (or in a population containing a best string if the number of best strings is more than one) for all $i \geq 2$.

4.2. Example

An example is provided here which demonstrates the convergence of EGA. In this example, a maximization problem is considered in the domain $D = \{0, 1, 2, 3\}$. We have considered $M = 2$, $l = 2$ and $A = \{0, 1\}$. The mating pool selection for the GA is performed here using Gen-Pool. The best string of the previous population is copied into the current one if the fitness values of all offspring are less than the previous best.

The strings representing x are $S_1 = 11$, $S_2 = 10$, $S_3 = 01$ and $S_4 = 00$. The fitness function values are taken to be

$$fit(S_1) = 1$$
$$fit(S_2) = 4$$
$$fit(S_3) = 2$$

and

$$fit(S_4) = 3.$$

The strings can be classified into four classes and are $\mathcal{S}_1 = \{S_2\}$, $\mathcal{S}_2 = \{S_4\}$, $\mathcal{S}_3 = \{S_3\}$ and $\mathcal{S}_4 = \{S_1\}$.

The number of populations or states is $\binom{2^2+2-1}{2} = 10$ and they are

$$Q_1 = \{10, 10\},\ Q_2 = \{10, 00\},\ Q_3 = \{10, 01\},\ Q_4 = \{10, 11\}.$$
$$Q_5 = \{00, 00\},\ Q_6 = \{00, 01\},\ Q_7 = \{00, 11\},\ Q_8 = \{01, 01\}.$$
$$Q_9 = \{01, 11\},\ Q_{10} = \{11, 11\}.$$

The partition over the populations is given below.

$$E_1 = \{Q_1, Q_2, Q_3, Q_4\},$$
$$E_2 = \{Q_5, Q_6, Q_7\},$$
$$E_3 = \{Q_8, Q_9\},$$
$$E_4 = \{Q_{10}\}.$$

We are representing here the transition probabilities as $p_{i,j}$ where $i, j = 1, 2, \ldots, 10$ for convenience. The n-step transition probability matrices for $n = 1$ and 1024, are given below for $p = 0.5$ and $q = 0.01$.

$$P^{(1)} = \begin{bmatrix}
0.960596 & 0.009999 & 0\,009803 & 0.019602 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.318435 & 0.667011 & 0.008056 & 0.006498 & 0 & 0 & 0 & 0 & 0 & 0 \\
0\,426975 & 0.228811 & 0.331134 & 0\,013080 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.617890 & 0.009608 & 0\,010230 & 0.362272 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.000098 & 0\,019406 & 0.000196 & 0.000002 & 0.960696 & 0\,019506 & 0.000196 & 0 & 0 & 0 \\
0.000036 & 0.007081 & 0.004760 & 0\,000048 & 0.350488 & 0\,632812 & 0\,004775 & 0 & 0 & 0 \\
0\,000098 & 0.014555 & 0.180271 & 0.004853 & 0\,540372 & 0.019506 & 0\,240345 & 0 & 0 & 0 \\
0.0 & 0.000002 & 0.000196 & 0\,000002 & 0\,000098 & 0.019406 & 0\,000196 & 0.960596 & 0.019504 & 0 \\
0\,000011 & 0.000045 & 0.004422 & 0\,002244 & 0.000044 & 0.008712 & 0.004422 & 0.431255 & 0.548845 & 0 \\
0.000098 & 0.000002 & 0.000196 & 0.019406 & 0 & 0.000002 & 0.000196 & 0.000098 & 0.019406 & 0.960596
\end{bmatrix}$$

$$P^{(1024)} = \begin{bmatrix}
0.918654 & 0.038293 & 0.014367 & 0.028922 & 0.000000 & 0.000000 & 0\,000000 & 0.0000000 & 0.000000 & 0\,000000 \\
0.918652 & 0.038292 & 0.014367 & 0.028922 & 0.000000 & 0.000000 & 0.000000 & 0.0000000 & 0.000000 & 0.000000 \\
0.918652 & 0.038292 & 0\,014367 & 0\,028922 & 0.000000 & 0.000000 & 0\,000000 & 0\,0000000 & 0.000000 & 0\,000000 \\
0.918651 & 0\,038292 & 0.014367 & 0.028922 & 0.000000 & 0\,000000 & 0\,000000 & 0\,0000000 & 0.000000 & 0.000000 \\
0.918646 & 0.038292 & 0.014367 & 0.028922 & 0.000000 & 0.000000 & 0\,000000 & 0.000000 & 0.000000 & 0\,000000 \\
0.918648 & 0.038292 & 0.014367 & 0.028922 & 0.000000 & 0.000000 & 0.000000 & 0.0000000 & 0.000000 & 0.000000 \\
0.918649 & 0\,038292 & 0.014367 & 0.028922 & 0.000000 & 0.000000 & 0.000000 & 0.0000000 & 0\,000000 & 0\,000000 \\
0.918628 & 0.038291 & 0\,014366 & 0.028922 & 0\,000000 & 0\,000000 & 0\,000000 & 0.0000000 & 0.000000 & 0.000000 \\
0.918632 & 0.038292 & 0.014367 & 0\,028922 & 0\,000000 & 0.000000 & 0.000000 & 0.0000000 & 0\,000000 & 0\,000000 \\
0.918631 & 0.038292 & 0\,014366 & 0\,028922 & 0\,000000 & 0\,000000 & 0\,000000 & 0.0000000 & 0\,000000 & 0\,000000
\end{bmatrix}$$

The figures in the above matrices are given up to 6 decimal places. It can be easily seen from the matrix $P^{(1024)}$ that $p_{i,j}^{(1024)}$ are nearly zero for all $j \geq 5$ and for all $i = 1, 2, \ldots, 10$. It can also be seen that all the rows are nearly identical in $P^{(1024)}$. From the literature on Markov chains,[7] it follows straightly that all the rows of $P^{(n)}$ are necessarily identical for sufficiently large n, i.e., $p_{i,j}^{(n)}$ are independent of i for sufficiently large n. This fact ensures the convergence of the GA for the considered model $p_{i,j}^{(1024)} \approx 0$, for $j \geq 5$ means that the probability of reaching the optimal solution is 1 after 1024 iterations. Thus, the process needs to evaluate 1024 populations (or 2048 strings) for convergence, though the problem under consideration needs the evaluation of only 4 strings.

Note that for the above demonstration with string length 2 and the populations size 2, the order of the transition probability matrix is 10. If we consider $l = 3$ and $M = 4$, the order will drastically increase to 330. Since it is very inconvenient to present such a huge matrix, the above mentioned problem with small values of string length and population size is considered as an illustration.

5. DISCUSSION AND CONCLUSIONS

The proof for convergence of EGA is provided when three basic genetic operations are performed to carry out the search. Mutation, which is generally considered as a secondary genetic operation,[5] turned out to be the most powerful tool in achieving the convergence of the process. Though we have performed single point crossover operation and incorporated Gen-Pool for generating mating pool, one can use any other variations of these operators available in the literature.[2,10] The convergence for GAs with other strategies of preserving the best string (within or outside the population) can be proved similarly.

The material presented here is similar to that of Ref. 6. The process has been viewed as a Markov chain in both works. We have incorporated the fitness function in the present Markov chain model. Secondly, the preservation of the best string has also been taken into consideration. Incorporation of these two concepts enables us to show the eventual convergence of the process to a global optimal string.

The proof provided in this article is not the panacea of all the problems in GAs. Termination rules which provide near optimality mathematically, are to be found. The search process in GAs should be such that the number of strings evaluated needs to be less than the total number of strings (i.e., a^L). Optimal values for M, p and q depend highly on the characteristics of the fitness function and need to be found for a given L. The utility of crossover operation needs to be investigated theoretically.

ACKNOWLEDGMENTS

The authors acknowledge Dr. M. K. Kundu and Dr. N. R. Pal for their helpful comments and discussions during the course of this work. This work was carried out while Prof. S. K. Pal held a Jawaharlal Nehru fellowship.

REFERENCES

1. C. A. Ankerbrandt, B. P. Buckles and F. E. Petry, "Scene recognition using genetic algorithms with semantic nets", *Pattern Recognition Letters* **11** (1990) 285–293.
2. R. Belew and L. Booker, eds., *Proc. 4th Int. Conf. on Genetic Algorithms*, San Diego, CA, 1991.
3. S. Bornholdt and D. Graudenz, "General asymptotic and neural networks and structure design by genetic algorithms", *Neural Networks* **5** (1992) 327–334.
4. R. Das and D. Whitley, "The only challenging problems are deceptive: Global search by solving order-1 hyperplanes", as in Ref. 2, pp. 166–173.
5. L. Davis, eds., *Genetic Algorithms and Simulated Annealing*, Pitman Publishing, 1987.
6. T. E. Davis and C. J. Principe, "A simulated annealing like convergence theory for the simple genetic algorithm", as in Ref. 2, pp. 174–181.
7. W. Feller, *An Introduction to Probability Theory and its Applications (Vol. I)*, Wiley Eastern Pvt. Ltd., New Delhi, 1972.
8. D. E. Goldberg, *Genetic Algorithms: Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, Inc., 1989.
9. M. Hulin, "Analysis of schema distribution", as in Ref. 2, pp. 190–196.
10. Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*, Springer Verlag, 1992.
11. S. K. Pal, D. Bhandari and M. K. Kundu, "Genetic algorithms for optimal image enhancement", *Pattern Recognition Letters* **15** (1994) 261–271.
12. N. J. Radcliffe, "Formal analysis and random respectful recombination", as in Ref. 2, pp. 222–229.
13. W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection", *Pattern Recognition Letters* **10** (1989) 335–347.
14. M. D. Vose, "Generalizing the notion of schema in genetic algorithms", *Artificial Intelligence* **50** (1988) 385–396.
15. D. Whitley, T. Starkweather and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity", *Parallel Computing* **14** (1990) 347–361.

Received 7 June 1994; Revised 11 September 1995.

Dinabandhu Bhandari received his M.Sc. in Applied Mathematics from the University of Calcutta in 1987, and M.Tech. in Computer Science from Indian Statistical Institute, Calcutta in 1990. He has been a Senior Research Fellow of the Indian Statistical Institute working in the Machine Intelligence Unit for his Ph.D. degree.

Currently, he is working in Tata Consultancy Services as a Senior System Analyst. His research interests include genetic algorithms, neural nets, fuzzy logic and pattern recognition.

C. A. Murthy received the B.Stat. (honors.), M.Stat., and Ph.D. degrees from Indian Statistical Institute, Calcutta, in 1979, 1980, and 1989, respectively.

He visited the Department of Computer Science of Michigan State University, East Lansing, as UNDP fellow in 1991-1992. His research interests include pattern recognition, image analysis, fuzzy sets, cluster analysis, fractals, and neural networks. He is an Associate Professor in the Machine Intelligence Unit at the Indian Statistical Institute and is currently visiting Pennsylvania State University, USA. Dr. Murthy is a member of the Indian Society for Fuzzy Mathematics and Information Processing (ISFUMIP) and the Indian Unit for Pattern Recognition and Artificial Intelligence (IUPRAI).

Sankar K. Pal is a Professor and Founding Head of Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He obtained M.Tech. and Ph.D. in Radiophysics and Electronics in 1974 and 1979, respectively, from the University of Calcutta, India. In 1982 he received another Ph.D. in Electrical Engineering along with DIC from Imperial College, University of London. In 1986 he was awarded a Fulbright Post-doctoral Visiting Fellowship to work at the University of California, Berkeley and the University of Maryland, College Park, U.S.A. In 1989 he received an *NRC-NASA Senior Research Award* to work at the NASA Johnson Space Center, Houston, Texas, USA.

He received the 1990 Shanti Swarup Bhatnagar Prize in Engineering Sciences, 1993 Jawaharlal Nehru Fellowship, 1993 Vikram Sarabhai Research Award, 1993 NASA Tech Brief Award, 1994 IEEE Transaction on Neural Networks outstanding paper Award and 1995 NASA Patent Application Award. His research interests mainly include pattern recognition, image processing, neural nets, genetic algorithms, and fuzzy sets and systems. He is a co-author of the book *Fuzzy Mathematical Approach to Pattern Recognition*, Wiley, 1986 and a co-editor of two books *Fuzzy Models for Pattern Recognition*, IEEE Press, 1992 and *Genetic Algorithms for Pattern Recognition*, CRC Press, 1996.

Prof. Pal is a *Fellow* of the IEEE, USA, Indian National Science Academy, Indian Academy of Sciences, National Academy of Sciences, India, Indian National Academy of Engineering, Institute of Engineers, India, and the IETE, a Member, *Executive Advisory Editorial Board, IEEE Trans. Fuzzy Systems and International Journal of Approximate Reasoning*, and an *Associate Editor, IEEE Trans. Neural Networks, Pattern Recognition Letters, Neurocomputing, Applied Intelligence, Information Sciences: Applications*, and *Far-East Journal of Mathematical Sciences*.