Genetic Algorithms: Stopping Times and A New Model

M.Tech Dissertation Report
by
Priyankar Ghosh

under supervision of

Prof. C. A. Murthy
Machine Intelligence Unit (MIU)
Indian Statistical Institute, Calcutta

July 22, 2000

Certificate of Approval

We the undersigned, hereby certify that Mr. Priyankar Ghosh of Indian Statistical Institute, Calcutta, has completed this dissertation on "Genetic Algorithms: Stopping Times and A New Model" as a part of the M.Tech(Computer Science) course, to our satisfaction.

External Examiner

Supervisor
Dr. C. A. Murthy

Acknowledgement

I would like to express my sincerest gratitude towards Dr. C. A. Murthy for providing me with a very interesting and challenging dissertation problem. He has helped me a lot and guided me with his brilliant ideas for which I was able to sail smoothly even in troubled waters. Also, I owe a lot to him for helping me in this report writing process.
I am obliged to my friends Mr. Sanjay Dutta, Mr. Sasthi Charan Ghosh for helping me prepare the LaTeX output of this dissertation.

With regards
Priyankar Ghosh
Priyankar Ghosh

Contents

Page No.

Chapter 1

Introduction

Mankind has been trying to solve optimization problems in different spheres of life. There have been several ways in which those problems have been attempted previously. Some of these are:

- Calculus based techniques: use indirect methods where a set of non-linear equations is solved or direct methods which are tools like hill climbing.
- Enumerative techniques: conceptually very simple, involving evaluation of objective function at every point of the search space.
- Random techniques: search space is investigated at random.

Genetic Algorithms(GAs) are stochastic search methods belonging to the last category, random techniques. They perform a multi-dimensional search in very large, complex and multimodal search spaces in providing an optimal solution for evaluation(*fitness*) function of an optimization problem. The idea of GAs originated from the principles of natural genetic systems, particularly from the theory of evolution. GAs are empirically found to provide global near optimal solutions to various complex optimization problems when the search space is discrete.

Genetic Algorithms came into existence since researchers wanted to simulate the natural genetic systems on computer. Applying GAs to solve optimization problems became a natural use of GAs since the human genetic system is supposed to be the best in the world and it evolved over various generations. While solving an optimization problem using GAs, each potential solution is encoded as a string(called "chromosome") of finite length (say, L) 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. Usually, three naturally occurring phenomena are incorporated in GAs. The three phenomena are

- Reproduction/selection
- Crossover
- Mutation.

The above biologically inspired phenomena are applied on the chromosomes to yield potentially better chromosomes. In each generation(mathematically "iteration") a new population of the same size is generated from the current population using the above phenomena and this new population is then used to generate another population. Note that the number of possible populations is finite as M, L,and $\mathcal{A}$ are finite.

Some of the application areas of GAs include operations research, pattern classification and feature selection, image processing[10] and scene recognition, rule generation and classifier systems[9], neural network design[8], scheduling problems, VLSI design, path planning[7] and the travelling salesman problem[6], graph coloring[5], numerical optimization.

There are several problems in applying genetic algorithms for real life applications. The problem of finding the stopping time for GAs and obtaining a new model in this regard are attempted here.

1.1 Description of Genetic Algorithms

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

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

Without loss of generality, we assume $f(x) > 0, \ \forall\, x \in D$. This can be achieved by using transformations. Since D is finite, D is a discrete domain. It can also be a subset of any finite dimensional space. In general, if δ is the number of parameters to be found using Gas and A_i represents the finite set of possible values of the i^{nth} parameter, then $D = A_1 \mathrm{x} A_2 \mathrm{x} \ldots \mathrm{x} A_\delta$.

Notations:

M: population size, a finite integer
L: string length, a finite integer
p: crossover probability, $p \in (0, 1)$
q: mutation probability, $q \in (0, 1)$

Each potential solution $x \in D$ is encoded as a string over a finite set of alphabet $\mathcal{A}$. This encoding is called chromosomal representation of solution x. Each string S corresponds to a value x in D and is of the form

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

The total number of strings i.e. the number of different values for the variable x is a^L. L is chosen such that $a^{L-1} < |D| \leq a^L$. It is also to be noted that the total number of strings(i.e. a^L) varies with L.

For simplicity, we take $\mathcal{A} = \{0, 1\}$, i.e. a binary string of length L is a chromosomal or string representation of a possible solution.

The evaluation function plays the role of environment and it ranks the solutions in terms of their fitness. The fitness function is chosen depending on the problem to be solved in such a way that the strings(possible solutions) representing good points in the search space have high fitness values. This is the the only information(also known as the *pay-off* function) that GAs use while searching possible solutions.

Here evaluation function or fitness function *fit* for a string S is equivalent to the function f i.e.

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

where the string S corresponds to x in D.

Each chromosome actually refers to a coded possible solution and is considered as an individual. A set of such chromosomes in a generation is called a population, the size of which may be constant or may vary from one generation to another. Here we keep the population size fixed throughout, though the assumption of fixed population size is not realistic in natural genetic systems. A genetic algorithm starts with a population of M potential sulutions in the form of chromosomes or strings. Each string of the population 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 the current population. Members of this population undergo reproduction by means of crossover and mutation operation to form a new population of solutions for the next iteration. A common practice is to choose the initial population randomly. Let, at iteration t, the population be

$$P^{(t)} = \{S_1^{(t)}, S_2^{(t)}, \cdots, S_M^{(t)}\}.$$

Now we discuss how the operations selection, crossover, mutation are performed on a population.

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. Here we consider the following strategy for generating mating pool.

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

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

- 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 are expected to die off. Several other strategies for the generation of mating pool are available in the literature.

crossover: The main purpose of crossover is to exchange information between randomly selected parent chromosomes by recombining parts of the corresponding strings. It recombines genetic material of two parent chromosomes to produce off-spring for the next generation. Single point crossover is one of the most commonly used schemes and between two chromosomes

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

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

it is performed in the followinh way.

Generate randomly an integer position *pos* from the range $\{1, 2, \cdots, L-1\}$. The two chromosomes β and γ are replaced by a pair β' and γ' where

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

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

Crossover operation on the mating pool of size M(M is even) can be performed in 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 to obtain $\frac{M}{2}$ pairs.

- Let p be the probability that a given pair of chromosomes take part in the crossover operation. For each pair of strings, generate a random number *rnd* from [0, 1]. If $rnd \leq p$, perform above crossover scheme beteen the pair of strings; otherwise no crossover is performed.

The crossover operation between two strings, as stated above, is performed at one position and that is why it is called single point crossover. Some other forms of crossover operation are also available in the literature.

Mutation: The main aim of mutation is to introduce genetic diversity into the population. sometimes, it helps to regain the information lost in earlier generations. Like natural genetic systems, mutation in GAs is usually performed occasionally.

Mutation is performed on each character bit of a string with probability $q(\gtrsim 0)$. 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 bit β_i at the i^{nth} 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 a randomly selected member from the set $(\mathcal{A} - \{\beta_i\})$.

Note that any string can be generated from any given string by mutation operation only. In case of binary chromosomal representation, mutation operation is performed by negating the bit value with probability q.

Alg-1: The basic steps of Gas are described below.

1. Generate an initial population Q of size M.
2. Construct a mating pool from Q using above mentioned selection operation. Perform crossover and mutation operations on the strings of the mating pool and obtain a population Q_{tmp}.
3. Calculate the fitness value of each string of Q_{tmp} and find the best string S_{best} of Q_{tmp}. If the best string is not unique, then call anyone of the best strings of Q_{tmp} as S_{best}.
4. Set $Q = Q_{tmp}$.
5. Go to step 2. ♠

Note that steps 2, 3 and 4 together make an iteration.

Note: Other models of Gas are variations of the above mentioned model. The variation may occur in the variations of the three operators, viz, selection, crossover and mutation or in the introduction of any other new operator. One such strategy is as follows. The knowledge about the best string obtained so far is usually preserved within the population. Genetic algorithms with this strategy i.e. maintaining the best string within the population are referred as *genetic algorithms with elitist model or EGA*. Other variations may occur in the choice of the alphabet $\mathcal{A}$ on which the strings are to be coded i.e. the chromosomal representation of strings may not necessarily be binary.

1.2 Issues regarding Gas

There are several issues to be resolved while applying genetic algorithms for real life applications. They are discussed below.

1.2.1 General Problems in applying Gas

It may be noted that no mention has been made about the number of iterations needed to stop the process in Gas(Alg-1) for obtaining optimal or near optimal solution. Usually, Gas run for a fixed number of iterations or terminate if no improvement is found for a fixed number of iterations. We don't know whether the Gas will converge or not, i.e. whether the Gas will produce the best string or not. That is, the convergence of Gas is not guaranteed. Gas have been successfully modelled as Markov Chains. Bhandari et. al.[2] preserved the knowledge of the previous best in their model and proved the convergence of Gas to optimal string. In their model, the knowledge about the best string obtained so far is usually preserved within the population. This strategy is called *elitism.* A way of implementing *elitism* is that the best string of the current population is copied into the the new population by replacing the worst if the fitness values of all individuals in the new population is less than the previous best. Genetic algorithms with this strategy i.e. maintaining the best string within the population are referred as *genetic algorithms with elitist model or EGA*. Henceforth, we will stick to *elitism* strategy to ensure the convergence.

The convergence of *EGA* has been proved(i.e. an *EGA* will result in the best string as the number of iterations n goes to ∞) mathematically. The literature on stopping times of EGAs is sparse and the existing stopping times are not satisfactory[2]. More specifically, the problem is to find a value for n which will provide the best solution.

Sometimes, it may happen that no improvement is found for many consecutive iterations, but we are not sure about the optimality of the best string obtained so far. We really don't know what move we should adopt to resolve this problem.

The values for the genetic parameters L, M, p and q have to be chosen properly before performing genetic operations. When the search space is continuous, we have to make it discrete by digitizing the continuous space and then we have to decide on the value of L and the alphabet $\mathcal{A}$ on which the strings will be coded. Higher value of

L corresponds to a bigger size of domain and hence the choice of L inherently provides a measure of precision for the solution. It has already been stated that M is taken as an even integer. No guidelines exists in the literature for choosing the appripriate value of M. It is not known to us how to select the crossover probability p and the mutation probability q. Since mutation occurs occasionally, it is clear that the probability of performing mutation operation will be very slow. A varying mutation probability with respect to iteration number may be useful in faster convergence of the process[2, 3]. we have no clear idea how to define the crossover and mutation operation in different problems.

1.2.2 Specific Problems attempted

In this work, we tried to find an ϵ-optimal stopping time for EGAs. In other words, we tried to find termination rules which provide near optimality mathematically. The work stated in this dissertation is based upon the concept of ϵ-optimal stopping time, defined theoretically for any EGA. In the literature, two types of stopping times are available—pessimistic and optimistic[2]. Pessimistic stopping time is derived with the assumption that every iteration starts with the worst population and thus the number of strings searched in this case is larger than the number of strings in the search space. Hence, this is not useful in practice. On the other hand, optimistic stopping time is derived with the assumption that the fitness function is well related with respect to the Hamming distance[2]. This assumption may not always be satisfied for an arbitrary fitness function. So, this is also not useful in practice. Also, we have proposed a new model of Gas in this regard so that the convergence is likely to be faster and it is likely to work for larger class of problems.

1.3 Data Sets used

For our experimental work with different models of genetic algorithms, the following fitness functions and data sets have been used.

<u>Function 1</u>

$$fit(x) = (k+1)Sin(2mx), \quad x \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$$

$$\text{where } k = 0, 1, 2, \ldots\ldots; \quad m = .5, 2, 8$$

we have taken the range of x as $[0,\ 32)$ in each of the following three cases and

we have digitized the interval into 2^L equally spaced points where L is the string length.
Function 1(a): For $m = .5$

$$\begin{aligned} fit_1(x) &= Sin(x) \quad , \quad x \in [0 \ , \ 2\pi] \\ &= 2Sin(x) \quad , \quad x \in [2\pi \ , \ 4\pi] \\ &= 3Sin(x) \quad , \quad x \in [4\pi \ , \ 6\pi] \\ &\vdots \end{aligned}$$

Here $L = 15$ and $M = 10$. $\quad Max\{fit_1\} = 5$

Function 1(b): For $m = 2$

$$\begin{aligned} fit_2(x) &= Sin(4x) \quad , \quad x \in \left[0 \ , \ \tfrac{\pi}{2}\right] \\ &= 2Sin(4x) \quad , \quad x \in \left[\tfrac{\pi}{2} \ , \ \pi\right] \\ &= 3Sin(4x) \quad , \quad x \in \left[\pi \ , \ \tfrac{3\pi}{2}\right] \\ &\vdots \end{aligned}$$

Here $L = 25$ and $M = 30$. $\quad Max\{fit_2\} = 21$

Function 1(c): For $m = 8$

$$\begin{aligned} fit_3(x) &= Sin(16x) \quad , \quad x \in \left[0 \ , \ \tfrac{\pi}{8}\right] \\ &= 2Sin(16x) \quad , \quad x \in \left[\tfrac{\pi}{8} \ , \ \tfrac{2\pi}{8}\right] \\ &= 3Sin(16x) \quad , \quad x \in \left[\tfrac{2\pi}{8} \ , \ \tfrac{3\pi}{8}\right] \\ &\vdots \end{aligned}$$

Here $L = 50$ and $M = 50$. $\quad Max\{fit_3\} = 82$

<u>Function 2</u>

$$fit_4(x) = xSin(\frac{1}{1-x}), \quad x \in [0\ ,\ 1)$$

Here $L = 50$. So, the number of digitized points= 2^{50} for the interval $[0\ ,\ 1)$.
$M = 50$. $Max\{fit_4\}$ is very close to one.

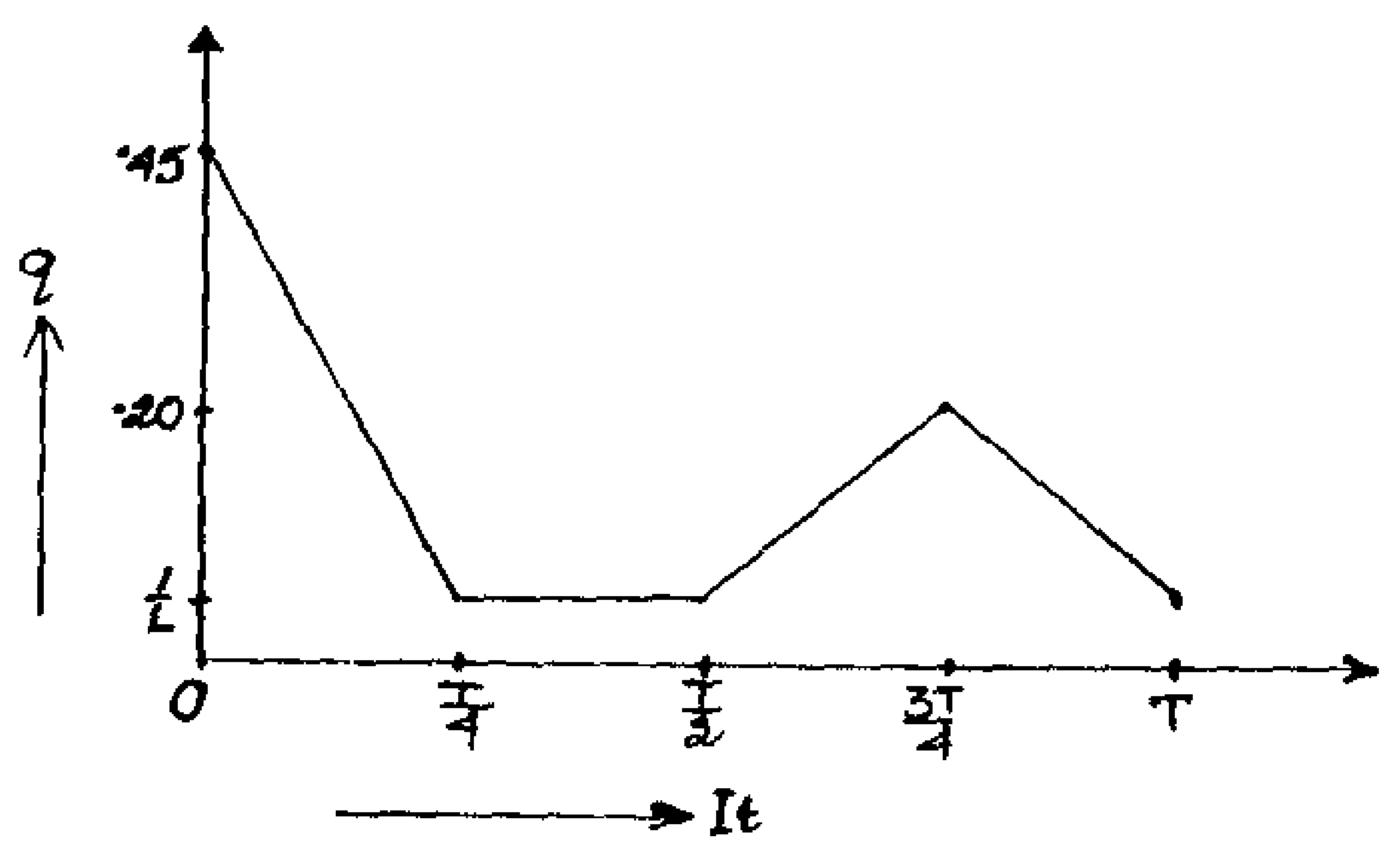
<u>Function 3</u> (a minimum deceptive problem)

$L = 50$ and the best string among the 2^{50} binary strings is $S^* = 111\ldots11(50\ times)$

$$fit_5(S^*) = L + 1$$
$$fit_5(S) = D(S, S^*),\ S \neq S^*$$

where Hamming distance $D(S, S^*)$ = number of bits at which the two strings S and S^* are different. Note that the function takes only $L+1$ values and $Max\{fit_5\} = 51$ in this case.

The utility of varying mutation probability with respect to the iteration number has already been stated in the previous section. In our experiment, we shall vary the mutation probability with respect to the iteration number as shown in the following figure:



where **q**: mutation probability, **It**: iteration number and **T**: total iterations. ♠

Chapter 2 deals with the stopping time for Mirror model[4] of EGAs and chapter 3 deals with a new proposed model of GAs.

Chapter 2

ϵ-Optimal Stopping Time for MEGA Model

The utility of optimal stopping times for genetic algorithms has already been stated in the previous chapter. This chapter deals with finding the ϵ-optimal stopping time of a particular model, namely Mirror model[4] of EGAs(call it MEGA model).

2.1 Description of MEGA Model

This model is applicable for binary chromosomal representation of strings. Here we use the word "complementation"— the complement of a string is a string obtained by negating each bit position of the given string. This model is based on elitism strategy also.

Alg-2: Basic Steps of the model:

1. Generate an initial population Q of size M by choosing $\frac{M}{2}$ strings randomly and taking the complementation of those randomly chosen $\frac{M}{2}$ strings, thus giving rise to M strings in total .
2. Calculate the fitness value of each string S of Q and find the best string S_{best} of Q. If the best string is not unique, then call any one of the best strings of Q as S_{best}.
3. Construct a mating pool from Q using the selection operation. Then perform the crossover operation on the strings of the mating pool. Construct a population

Q_{tmp} of size M by taking and complementing the first $\frac{M}{2}$ best strings among the generated strings after crossover operation.

4. Perform mutation operation on the strings of Q_{tmp} to give rise to a population Q'.

5. If the fitness value of each string of Q' is less than $fit(S_{best})$, then replace the worst string(may be any string) of Q' with S_{best}; otherwise, no replacement takes place in Q'. Then construct a new population Q'' of size M by taking and complementing the first $\frac{M}{2}$ best strings of Q'.

6. Set $Q = Q''$. Go to step 2. ♠

Rest of this chapter deals with finding the ϵ-optimal stopping time for the above mentioned model of Gas. The proof of convergence of EGA has been provided in [1]. The above model is a variation of EGA model. An EGA will result in the best string as the number of iterations n goes to ∞. But the process needs to be stopped for some finite value of n. The value of n at which the process is stopped is called the stopping time of the process. The objective here is to determine the value of n which is in some sense 'optimal'. More specifically, the problem is to find a value for n which will provide the best solution.

Note that, for starting population Q, Gas can result in many populations after n iterations. Thus, if the process is to be stopped after a specific number of iterations, it may not always be guaranteed that optimal string is obtained. The following lemma[2] stated below indicates that there is a positive probability of not obtaining the optimal solution after any finite number of iterations.

Lemma: Let the fitness function is such that there exists a string S_0 which is not an optimal string (there always exists such an S_0 as the function is not constant). Let us consider the population Q such that Q contains M copies of S_0. Then the probability that the process will remain in Q after n iterations is positive(>0) for all n.

So, it follows from the above lemma that no finite stopping time can guarantee the optimal solution. On the other hand, the process is to be terminated after finitely many iterations with the expectation that the process has achieved the optimal solution. Thus any decision regarding the stopping time should necessarily be probabilistic, since GA is a stochastic process. Now we discuss the meaning of ϵ-optimal stopping time below in the context of Gas.

2.2 Meaning and Definition of ϵ-Optimal Stopping Time

In this section we shall provide the definition of ϵ-optimal stopping time[2]. Genetic algorithms search over a space $\mathcal{S}$ of 2^L strings and eventually provide the best string with respect to the fitness function fit.

Let $\mathcal{C} = \{fit(S) \ : \ S \in \mathcal{S}\}$ and the number of elements in $\mathcal{C}$ be s ($s \leq 2^L$). Then $\mathcal{C}$ can be written as

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

Let us also assume, without loss of generality, that $F_s > 0$. 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. Therefore, the total number of distinct populations is

$$\mathcal{N} = \binom{2^L + M - 1}{M}$$

and let $\mathcal{Q}$ be the collection of all such distinct populations.

Fitness function value of a population Q is defined as

$$fit(Q) = Max_{S \in Q} fit(S), \quad Q \in \mathcal{Q}.$$

Then the populations can be partitioned into s sets E_i where

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

Let e_i be the number of elements in E_i, $i = 1, 2, \ldots, s$. In an iteration or generation, the genetic operators(selection, crossover, mutation) create a population $Q_{kl} \in E_k$, $l = 1, 2, \ldots, e_k$ and $k = 1, 2, \ldots, s$ from some $Q_{ij} \in E_i$. This generation of a population Q_{kl} from Q_{ij} is considered as a transition from Q_{ij} to Q_{kl}.

Let $p_{ij.kl}$ denote the probability that the genetic operators result in the population $Q_{kl} \in E_k$ from $Q_{ij} \in E_i$ where $j = 1, 2, \ldots, e_i$; $l = 1, 2, \ldots, e_k$ and $i, k = 1, 2, \ldots, s$. Then the probability of transition from Q_{ij} to any population in E_k can be calculated as

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

Clearly, by construction, for all $j = 1, 2, \ldots, e_i$ and $i = 1, 2, \ldots, s$

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

as we are following the elitism strategy.

Let $p_{ij,kl}^{(n)}$ be the probability that GA results in Q_{kl} after n iterations given that the initial population is Q_{ij} and let $p_{ij,k}^{(n)}$ denote the probability of reaching one of the populations in E_k from Q_{ij} at the n_{nth} iteration. Then

$$p_{ij.k}^{(n)} = \sum_{l=1}^{e_k} p_{ij.kl}^{(n)} \ .$$

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

Now, we are in a state for defining ϵ-optimal stopping time.

Let $Q_{ij} \in E_i$ and $Q_{ij}^{(n)}$ denote the population that is obtained after n iterations of the genetic algorithms with the starting population as Q_{ij} . Let $g_{ij}^{(n)}$ denote the fitness function value of the population $Q_{ij}^{(n)}$. Let $\mathcal{E}(g_{ij}^{(n)})$ denote the expected value of $g_{ij}^{(n)}$. Then

$$\mathcal{E}(g_{ij}^{(n)}) = \sum_{k=1}^{i} F_k p_{ij.k}^{(n)} \ .$$

Let $\epsilon \geq 0$ and N_0 be a positive integer. Then N_0 is said to be an ϵ-optimal stopping time for a GA if

$$n \geq N_0 \quad \Rightarrow \quad \mathcal{E}(g_{ij}^{(n)}) \geq F_1 - \epsilon \ , \quad \forall \, i,j$$

In particular, if $\epsilon = 0$, $N_0$is called 0-optimal stopping time or simply optimal stopping time. Note that, the above the definition can not be used directly for Gas as we don't know the value of F_1. So, we use the following.

Definition(ϵ-optimal stopping time) : Let $\epsilon \geq 0$ and N be a positive integer. Then N_0 is said to be an ϵ-optimal stopping time for Gas if

$$n \geq N \quad \Rightarrow \quad \mathcal{E}(g_{ij}^{(n)}) \geq F_1(1-\epsilon) \ , \quad \forall \, i,j \tag{1}$$

Note that F_1 is used in the above definition too. But the following manipulations remove the dependency of F_1 on the ϵ-optimal stopping time. Clearly,

$$\mathcal{E}(g_{ij}^{(n)}) = \textstyle\sum_{k=1}^{i} F_k p_{ij.k}^{(n)} \ \geq \ F_1 p_{ij.1}^{(n)} \ , \quad \forall \, i,j \tag{2}$$

From (1) and (2), N needs to be found such that

$$n \geq N \quad \Rightarrow \quad p_{ij}^{(n)} \geq 1 - \epsilon \ , \quad \forall \, i,j \ .$$

Note that, if N is an ϵ-optimal stopping time for a GA, then any $N_1 > N$ is also an ϵ-optimal stopping time for the GA. Thus, for a given $\epsilon > 0$, we want to find the minimal ϵ-optimal stopping time for GA, i.e. we want to find N_0 such that, if N_0 is an ϵ-optimal stopping time, then any $N < N_0$ would imply N is not an ϵ-optimal stopping time for the GA.

Now, the problem is to find the values for $p_{ij.1}^{(n)}$ for any initial population Q_{ij} .

2.3 Finding the actual ϵ-Optimal Stopping Time for MEGA Model

In [2], it has been proved that the probabilities of reaching to a population containing one of the best strings will be higher if the number of strings having the highest fitness function value is more. Let f_1 be a fitness function, which assumes distinct values $F_1, F_2, \cdots F_s$ defined on $\mathcal{S}$. Let S_1 and S_2 be two different strings such that

$$f_1(S_1) = f_1(S_2) \text{ and } f_1(S) \leq f_1(S_1), \ \forall S \in \mathcal{S}.$$

Let us now define f_2 on $\mathcal{S}$ as follows :

$$f_2(S) = f_1(S), \ \forall S \neq S_2 \text{ and } f_2(S_2) = F_i \text{ for some } i > 1.$$

Then ϵ-optimal stopping time for the function f_2 is also an ϵ-optimal stopping time for f_1. Thus, in this section, we deal with the functions which possess exactly one optimal string. Such fitness functions are termed as single optimal fitness function.

In the MEGA model, every iteration starts with a population of size M in which $\frac{M}{2}$ strings are complementary strings of the rest $\frac{M}{2}$ strings. Let $S^* \in \mathcal{S}$ be the optimal string for a single optimal fitness function, i.e.

$$fit(S^*) > fit(S), \ \forall S \in \mathcal{S}, \ S \neq S^*.$$

Let us consider a population Q' of size M such that

1. $\frac{M}{2}$ strings are complementary strings of the rest $\frac{M}{2}$ strings.
2. Hamming distance $D(S, S^*) = \frac{L}{2}, \ \forall S \in Q'$.

In any population having only property(1), the M strings can be paired into $\frac{M}{2}$ pairs, each pair contains the complementary string of the other string in the pair.

Let A_i be the event that at least one of the strings in the i^{th} pair is transformed into the optimal string S^* with mutation operation only, $i = 1, 2, \cdots, \frac{M}{2}$. Let us also assume that one string in a pair has Hamming distance x $(0 \leq x \leq L)$ from the optimal string S^* and hence the other string in the pair will have Hamming distance $L - x$ from the optimal string S^*. From the probability theory,

$$\begin{aligned} P(A_i) &= q^x(1-q)^{L-x} + q^{L-x}(1-q)^x - q^x(1-q)^{L-x}.q^{L-x}(1-q)^x \\ &= q^x(1-q)^{L-x} + q^{L-x}(1-q)^x - q^L(1-q)^L \end{aligned}$$

Hence $P(A_i)$ will be minimum when the function

$$f(x) = q^x(1-q)^{L-x} + q^{L-x}(1-q)^x$$

is minimum, as the term $q^L(1-q)^L$ is independent of x.

Lemma : $P(A_i)$ is minimum for $x = \frac{L}{2}$.

Proof : We shall minimize $f(x)$ where

$$f(x) = q^x(1-q)^{L-x} + q^{L-x}(1-q)^x$$

$$\begin{aligned} So,\ \tfrac{d}{dx}f(x) &= \tfrac{d}{dx}q^x(1-q)^{L-x} + \tfrac{d}{dx}q^{L-x}(1-q)^x \\ &= q^x(1-q)^{L-x}\{\log(q) - \log(1-q)\} + q^{L-x}(1-q)^x\{\log(1-q) - \log(q)\} \\ &= \{\log(q) - \log(1-q)\}\{q^x(1-q)^{L-x} - q^{L-x}(1-q)^x\} \end{aligned}$$

Generally, mutation probability q lie between 0.0 and 0.5 , i.e. $q < 1-q$ which implies

$$\log(q) - \log(1-q) < 0$$

Observations:

1. $For\ x = \frac{L}{2}$, $q^x(1-q)^{L-x} = q^{L-x}(1-q)^x \Rightarrow \frac{d}{dx}f(x) = 0.$
2. $For\ x < \frac{L}{2}$, $q^x(1-q)^{L-x} > q^{L-x}(1-q)^x \Rightarrow \frac{d}{dx}f(x) < 0.$
3. $For\ x > \frac{L}{2}$, $q^x(1-q)^{L-x} < q^{L-x}(1-q)^x \Rightarrow \frac{d}{dx}f(x) > 0.$

Clearly, at $x = \frac{L}{2}$, $f(x)$ is minimum. Hence, $P(A_i)$ is minimum when one of the strings in the i^{th} pair has Hamming distance $\frac{L}{2}$ from the optimal string.

Theorem : Let $s_{Q.1}$ represent the probability of reaching a population containing S^* from any arbitrary population Q having property(1) in one iteration with mutation operation alone. Then

$$s_{Q.1} \geq s_{Q'.1}\ ,\ \forall Q$$

Proof : From the principles of probability theory,

$$\begin{aligned} s_{Q.1} &= P(\bigcup_{i=1}^{\frac{M}{2}} A_i) \\ &= P(A_i \bigcup (\bigcup_{j \neq i} A_j)) \\ &= P(A_i) + P(\bigcup_{j \neq i} A_j) - P(A_i \bigcap (\bigcup_{j \neq i} A_j)) \\ &= P(A_i) + P(\bigcup_{j \neq i} A_j) - P(A_i) P(\bigcup_{j \neq i} A_j) \\ &= P(A_i)\{1 - P(\bigcup_{j \neq i} A_j)\} + P(\bigcup_{j \neq i} A_j) \end{aligned}$$

Note that the events A_i , $i = 1, 2, \cdots, \frac{M}{2}$ are independent. Now, if we consider $P(\bigcup_{j \neq i} A_j)$ is constant, then $s_{Q.1}$ will be minimum when $P(A_i)$ is minimum, i.e. when one of the strings in the i^{th} pair has Hamming distance $\frac{L}{2}$ from the optimal string.

Thus, $s_{Q.1}$ will be minimum when $Q = Q'$, i.e. $s_{Q.1} \geq s_{Q'.1}$, $\forall\, Q$.

Theorem : Let $p_{Q.1}$ be the probability of reaching a population containing the optimal string S^* from any arbitrary population Q in one iteration using selection, crossover and mutation operations in the MEGA model of GAs. Then

$$p_{Q.1} \geq s_{Q.1} , \ \forall\, Q$$

Proof : Let $Q = \{S_1, S_2, \cdots, S_M\}$ be the given population and let ρ_i $(i = 1, 2, \cdots, M)$ represent the probability of selecting the string S_i to the mating pool. Then

$$0 < \rho_i < 1 \ \ and \ \ \sum_{i=1}^{M} \rho_i = 1 .$$

Let, after selection, the possible number of mating pools be τ and they may be represented by Q_1, Q_2, $\cdots, Q_\tau$.

Note that the probability of obtaining a population $Q_i = \{S_{i1}, S_{i2}, \cdots, S_{iM}\}$ where $S_{ij} \in Q$, $\forall\, i, j$ is

$$= \prod_{j=1}^{M} \rho_{ij} , \ \ \forall\, i = 1, 2, \cdots, \tau .$$

Then, $\sum_{i=1}^{\tau} \prod_{j=1}^{M} \rho_{ij} = 1$.

Given any mating pool Q_i , $1 \leq i \leq \tau$, the possible number of populations those may be obtained after crossover operation be V_i and they may be represented by $Q_{i1}, Q_{i2}, \cdots, Q_{iV_i}$. Let the probability of obtaining the population Q_{ij} from Q_i by using the crossover operation alone be represented by W_{ij} where

$$0 < W_{ij} < 1 \ , \quad \forall\, j = 1, 2, \cdots, V_i \ \text{ and } \ i = 1, 2, \cdots, \tau$$

$$\text{and } \ \sum_{j=1}^{V_i} W_{ij} = 1 \ , \quad \forall\, i = 1, 2, \cdots, \tau$$

After the crossover operation, the population Q_{ij} will be transformed into a population Q'_{ij} where $\frac{M}{2}$ strings are complementary strings of the rest $\frac{M}{2}$ strings.

Then the possible populations that may be obtained after selection and crossover operation are Q'_{ij} , $j = 1, 2, \cdots, V_i$; $i = 1, 2, \cdots, \tau$ with probabilities $W_{ij} \prod_{k=1}^{M} \rho_{ik}$.

Note that, $\sum_{i=1}^{\tau} \sum_{j=1}^{V_i} W_{ij} \prod_{k=1}^{M} \rho_{ik} = 1$.

Then

$$\begin{aligned}
p_{Q.1} &= \sum_{i=1}^{\tau} \sum_{j=1}^{V_i} W_{ij} \prod_{k=1}^{M} \rho_{ik} s_{Q'_{ij}.1} \\
&\geq \sum_{i=1}^{\tau} \sum_{j=1}^{V_i} W_{ij} \prod_{k=1}^{M} \rho_{ik} s_{Q'.1} \\
&= s_{Q'.1} \sum_{i=1}^{\tau} \sum_{j=1}^{V_i} W_{ij} \prod_{k=1}^{M} \rho_{ik} \\
&= s_{Q'.1}
\end{aligned}$$

Now we can compute $s_{Q'.1}$. Let us denote $\alpha = q^{\frac{k}{2}}(1-q)^{\frac{k}{2}}$. From the probability theory

$$\begin{aligned}
s_{Q'.1} &= \binom{M}{1}\alpha - \binom{M}{2}\alpha^2 + \binom{M}{3}\alpha^3 - \cdots\cdots + (-1)^{M-1}\binom{M}{M}\alpha^M \\
&= \Sigma_{r=1}^{M}\binom{M}{r}\alpha^r(-1)^{r-1} \\
&= 1-(1-\alpha)^M \\
&= 1-\left(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}}\right)^M
\end{aligned}$$

Note that the probability of not reaching a population containing the optimal string S^* in n iterations from an initial population Q is $(1-p_{Q.1})^n$.

Now $\quad p_{Q.1} \geq s_{Q'.1} \Rightarrow (1-p_{Q.1})^n \leq (1-s_{Q'.1})^n$.

Let $0 < \epsilon < 1$. The minimum value of n for which $(1-s_{Q'.1})^n < \epsilon$ gives an upper bound for ϵ-optimal stopping time for MEGA model of GAs.

Now

$$\begin{aligned}
&(1-s_{Q'.1})^n < \epsilon \\
\Rightarrow\ &\left(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}}\right)^{Mn} < \epsilon \\
\Rightarrow\ &\left(\frac{1}{1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}}}\right)^{Mn} > \frac{1}{\epsilon} \\
\Rightarrow\ &n > \frac{\log\frac{1}{\epsilon}}{M\log\left(\frac{1}{1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}}}\right)} = \frac{\log\epsilon}{M\log(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}})}
\end{aligned}$$

Let $\quad N(\epsilon, M, q, L) = \frac{\log\epsilon}{M\log(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}})}$

Note that, given ϵ, M, q and L, $N(\epsilon, M, q, L)$ is an upper bound for ϵ-optimal stopping time by construction. Since we have no knowledge of minimal ϵ-optimal stopping time, in the subsequent analysis we assume that $N(\epsilon, M, q, L)$ iterations will be performed during the process.

Remarks :

1. For a given ϵ, M, q and L, $N(\epsilon, M, q, L)$ is independent of the characteristics of the fitness function, crossover probability and the selection procedure.
2. Note that, if the starting population is not Q', then also $N(\epsilon, M, q, L)$ many iteration will be sufficient.
3. Given ϵ, q and L, $N(\epsilon, M, q, L)M$ (product of $N(\epsilon, M, q, L)$ and M) is a constant. Note that, in each iteration of MEGA model, $2M$ strings are evaluated. So, the number of strings searched upto $N(\epsilon, M, q, L)$ many iterations is $2MN(\epsilon, M, q, L)$ which is also a constant, independent of population size M.
4. It can be seen that, for given M, q, L

   $$\epsilon_1 < \epsilon_2 \ \Rightarrow \ N(\epsilon_1, M, q, L) \ > \ N(\epsilon_2, M, q, L).$$

   It implies that the number of iterations required is more to obtain a more accurate solution.
5. It is also clear that, given ϵ, q, M

   $$L_1 > L_2 \ \Rightarrow \ N(\epsilon, M, q, L_1) \ > \ N(\epsilon, M, q, L_2).$$

   This also coincides with our intuition that, for a fixed ϵ, if the length of the strings increases, the required number of iterations also increases.

From the literature[2], the pessimistic ϵ-optimal stopping time is

$$N_{pes}(\epsilon, M, q, L) \ = \ \tfrac{\log \epsilon}{M \log(1-q^L)}$$

Note that

$$N(\epsilon, M, q, L) \ < \ N_{pes}(\epsilon, M, q, L) \qquad \text{for } \ q < 0.5$$

as

$$q < 1-q \quad \Rightarrow \quad q^{\frac{L}{2}} < (1-q)^{\frac{L}{2}} \quad \Rightarrow \quad q^{L} < q^{\frac{L}{2}}(1-q)^{\frac{L}{2}}$$

For pessimistic ϵ-optimal stopping time, the number of strings searched upto $N_{pes}(\epsilon, M, q, L)$ iterations is

$$\frac{\log \epsilon}{\log(1-q^L)}$$

and for our ϵ-optimal stopping time, the number of strings searched upto $N(\epsilon, M, q, L)$ iterations is

$$\frac{2\log \epsilon}{\log(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}})}.$$

Now

$$\frac{2\log \epsilon}{\log(1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}})} < \frac{\log \epsilon}{\log(1-q^L)}$$

holds

$$\begin{aligned} \textit{if} \quad 1-q^{\frac{L}{2}}(1-q)^{\frac{L}{2}} &< (1-q^L)^2 = 1-2q^L+q^{2L} \\ \textit{i.e. if} \quad q^L(1-\tfrac{1}{2}q^L) &< \tfrac{1}{2}q^{\frac{L}{2}}(1-q)^{\frac{L}{2}} \\ \textit{i.e. if} \quad (\tfrac{q}{1-q})^{\frac{L}{2}}(1-\tfrac{1}{2}q^L) &< \tfrac{1}{2} \end{aligned}$$

which holds good for moderately large values of L for $q < \frac{1}{2}$. Hence, the number of strings searched upto $N(\epsilon, M, q, L)$ iterations is also less for MEGA model.

2.4 Experimental Results

We have experimented the MEGA model with the fitness functions, as given in section 1.3, once with fixed mutation probability($q = 0.2$) and the other with varying mutation probability. The experimental results are given below.

fitness function	actual optimum value	number of iterations	value obtained from MEGA with fixed mute. prob.	value obtained from MEGA with varying mute.prob.
fit_1	5.00	50	4.999930	4.999999
			4.999984	4.999999
			4.999995	4.999999
			4.999953	5.000000
			4.999994	4.999999
fit_1	5.00	100	4.999999	5.000000
			5.000000	5.000000
			4.999999	5.000000
			4.999999	4.999999
			5.000000	5.000000
fit_2	21.00	100	20.999984	21.000000
			20.996787	21.000000
			21.000000	21.000000
			20.999651	20.999999
			20.999992	21.000000
fit_2	21.00	200	20.999954	21.000000
			21.000000	21.000000
			20.999995	21.000000
			20.999999	21.000000
			21.000000	21.000000
fit_3	82.00	200	81.996205	82.000000
			81.999983	82.000000
			81.998091	82.000000
			81.999990	82.000000
			81.999959	82.000000
fit_3	82.00	500	81.999958	82.000000
			82.000000	82.000000
			82.000000	82.000000
			81.999959	82.000000
			81.996550	82.000000

fitness function	actual optimum value	number of iterations	value obtained from MEGA with fixed mute. prob.	value obtained from MEGA with varying mute.prob.
fit_4	very close to 1	500	0.998853	0.999858
			0.998305	0.999914
			0.999377	0.999646
			0.998439	0.999883
			0.998273	0.999749
fit_4	very close to 1	2000	0.999634	0.999979
			0.999526	0.999907
			0.999567	0.999995
			0.999859	0.999927
			0.999096	0.999903
fit_5	51	200	41	51
			42	51
			39	51
			41	51
			40	49
fit_5	51	500	42	51
			43	51
			42	51
			40	51
			41	51
fit_5	51	5000	46	51
			44	51
			46	51
			45	51
			47	51

The above experimental results reflect the fact that the varying mutation probability is useful in faster convergence and the model works for a larger class of problems, even for minimum deceptive problems.

Chapter 3

A New Proposed Model of EGA

We have proposed a new model of genetic algorithms, which will be described and discussed in this chapter. Our expectation to this model is that it will converge to optimal string faster and work for a large class of problems, even for minimum deceptive problems.

3.1 Description of the New Model

This model is applicable when the chromosomal representation of strings is binary. We also follow elitism strategy in this model.

Alg-3: Basic steps are:

1. Generate an initial population Q of size M by choosing M strings randomly from the search space.
2. Calculate the fitness value of each string S of Q and find the best string S_{best} of Q. If the best string is not unique, then call any one of the best strings of Q as S_{best}.
3. Construct a mating pool from Q using selection operation. Then perform crossover operation on the strings of the mating pool and obtain a population Q_{tmp} of size M.
4. Construct a population Q' of size $2M$ comprising of all the M strings of Q and all the M strings of Q', *i.e.* $Q' = Q \cup Q_{tmp}$.

5. Perform mutation operation on the $2M$ strings of Q' to give rise to a population Q''.

6. If each string of Q'' has fitness value less than $fit(S_{best})$, then replace the worst(may be any) string of Q'' with S_{best}; otherwise, no replacement takes place in Q''.

7. Construct a new population by taking and complementing the first $\frac{M}{4}$ strings of Q'', thus giving rise to $\frac{M}{2}$ strings. Then choose another $\frac{M}{2}$ strings randomly from the search space. Call this population of size M as Q^*.

8. Set $Q = Q^*$.

9. Go to step 2. ♠

Note: Steps from 2 to 8 together make an iteration. Steps 6 and 7 can be implemented in such a way that in each iteration we evaluate only $2M$ strings.

3.2 Rationale behind the Model

In the above model, we are incorporating the effects of

- Elitism strategy
- Selection, crossover and mutation operations
- Random method

For mutation operation, we are considering the srings generated after the crossover operation and the strings of the current population. "Complementation" technique also generates new strings which will be useful for minimum deceptive problems, as we shall have no idea about the fitness functions while applying GAs. The strings generated by complementaion technique may have lower fitness values and hence, by selection operation, they might die off. That is why we are considering those strings directly for the mutation operation.

Also, note that the crossover operation may change the strings radically and new strings are generated. It may be the case that in the current population some strings are near to the optimal strings, but they differ from an optimal string by only a very few bit positions. So, in this case, the crossover operation on these strings may create problems. This is also the reason for considering the strings of the current population for mutation operation. Though the selction and crossover operations generate new strings for the next population, we are not sure about their performance with respect to the convergence of GAs to an optimal string.

3.3 Experimental Results

We have applied both the ordinary model of EGAs and our new model of EGAs on the fitness functions, as given in section 1.3, with varying mutation probability. We present below a comparison between the two models of EGAs with respect to their performances.

fitness function	actual optimum value	number of iterations	value obtained from ordinary model of EGAs with varying mute. prob.	value obtained from New Model of EGAs with varying mute. prob.
fit_1	5.00	25	4.996295	4.999994
			4.989751	4.999995
			4.992678	5.000000
			4.983751	5.000000
			4.993158	4.999999
fit_1	5.00	50	4.999950	5.000000
			4.999191	4.999999
			4.994375	5.000000
			4.998578	5.000000
			4.999276	5.000000
fit_2	21.00	100	20.999983	21.000000
			20.999959	21.000000
			20.999324	21.000000
			21.000000	21.000000
			20.999883	21.000000
fit_2	21.00	200	20.999993	21.000000
			21.000000	21.000000
			20.999976	21.000000
			20.999902	21.000000
			21.000000	21.000000
fit_3	82.00	200	81.993959	82.000000
			81.999760	82.000000
			81.999651	82.000000
			81.992735	82.000000
			81.997697	82.000000
fit_3	82.00	500	81.999959	82.000000
			81.999960	82.000000
			82.000000	82.000000
			81.999997	82.000000
			81.999951	82.000000

fitness function	actual optimum value	number of iterations	value obtained from ordinary model of EGAs with varying mute. prob.	value obtained from New Model of EGAs with varying mute. prob.
fit_4	very close to 1	500	0.988305	0.999646
			0.998273	0.999586
			0.997352	0.999863
			0.989325	0.999952
			0.999449	0.999842
fit_4	very close to 1	2000	0.999863	0.999997
			0.999799	0.999999
			0.999954	0.999988
			0.999980	0.999942
			0.999899	0.999987
fit_5	51	100	36	51
			34	51
			38	51
			40	49
			35	51
fit_5	51	200	41	51
			37	51
			39	51
			40	51
			42	51
fit_5	51	5000	50	51
			49	51
			50	51
			50	51
			50	51

The experimental results reflect the fact that the convergence of our new model is faster. The number of strings evaluated in each iteration of the new model is twice the population size. In spite of that, the new model is superior to the ordinary model of EGAs with respect to the performance. Also, the new model works very good for $fit_5()$ which coincides with our intuition that this new model will also work for a larger class of problems, even for minimum deceptive problems.

Chapter 4

Conclusion, Discussion and Scope for further work

In this dissertation work, the problem of finding termination rules mathematically, which provide near optimality, for elitist model of genetic algorithms has been attempted and we have succeeded to find an ϵ-optimal stopping time for Mirror model of EGAs(MEGA) from the pessimistic point of view as we have assumed in the derivation of ϵ-optimal stopping time that the process starts with the worst population. Here binary representation of chromosomes is discussed and we have explored the role of complemented strings in this model. We have also shown that this ϵ-optimal stopping time is better than the existing pessimistic ϵ-optimal stopping time. The experimental results reflect that the varying mutation probability with respect to the iteration number is useful in faster convergence of the process.

We also tried to find a new model of EGAs, whose convergence would be faster, though the convergence of EGAs has already been proved. So, we tried to improve upon the ordinary model of EGAs and a new model came out. The rationale behind our new model has been discussed. The experimental results show the coincidence with our intuition that the convergence of this new model of EGAs is faster and the model works for a larger class of problems, even for minimum deceptive problems.

Further mathematical investigations are necessary to provide more general and realistic versions of stopping times for EGAs. Investigations are also necessary to judge theoretically the effect of varying mutation probability on stopping times as varying mutation probability with respect to the iteration number may be useful for faster convergence of the process. We have proposed the new model, but we were not able to find a good stopping time for this model. Here, the mathematical investigations are also necessary to judge the performances of the new proposed model of EGAs.

Bibliography

[1] D. Bhandari, C. A. Murthy, and S. K. Pal, "Genetic algorithms with elitist model and its convergence", *Int. J. of Pattern Recog. and Art. Intell.*, vol. 10, pp. 731-747, 1996.

[2] C. A. Murthy, D. Bhandari, and S. K. Pal, "ϵ-Optimal Stopping Time for Genetic Algorithms", *Fundamental Informaticae*, vol. 35, pp. 91-111, 1998.

[3] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms", *Pattern Recog. Lett.* , vol. 17, pp. 825-832, 1996.

[4] Rajeev Ayyagari, "ϵ-optimal stopping times for GAs", M.Stat II dissertation, Indian Statistical Institute, 1998-99.

[5] L. davis, ed., Handbook of Genetic Algorithm. New York: Van Nostrand Reinhold, 1991.

[6] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic algorithms for the travelling salesman problem", *Proc. 1st Int. Conf. Genetic Algorithms*, pp. 160-168, Hillsdale: Lawrence Erlbaum Associates, 1985.

[7] T. Cleghorn, P. Baffes, and L. Wang, "Robot path planning using a genetic algorithm", *Proc. SOAR(Houston)*, pp. 81-87, 1988.

[8] S. Bornholdt and D. Grandenz, "General asymmetric neural networks and structure design by genetic algorithms", *Neural Networks*, vol. 5, pp. 327-334, 1992.

[9] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms", *Art. Intell.*, vol. 40, pp. 235-282, 1989.

[10] A. Hill and C. J. Taylor, "Model-based image interpretation using genetic algorithms", *Image and Vision comput.*, vol. 10, pp. 295-300, 1992.