Fundamenta Informaticae 120 (2012) 145–164
DOI 10.3233/FI-2012-754
IOS Press

Variance as a Stopping Criterion for Genetic Algorithms with Elitist Model

Dinabandhu Bhandari, C. A. Murthy, Sankar K. Pal[*]
Center for Soft Computing Research
Indian Statistical Institute
Kolkata 700108, India
dinabandhu.bhandari@gmail.com, murthy@isical.ac.in, sankar@isical.ac.in

Abstract. Genetic Algorithm (GA) has now become one of the leading mechanisms in providing solution to complex optimization problems. Although widely used, there are very few theoretical guidelines for determining when to stop the algorithm. This article establishes theoretically that the variance of the best fitness values obtained in the iterations can be considered as a measure to decide the termination criterion of a GA with elitist model (EGA). The criterion automatically takes into account the inherent characteristics of the objective function. Implementation issues of the proposed stopping criterion are explained. Its difference with some other stopping criteria is also critically analyzed.

Keywords: Genetic Algorithm with Elitist Model, Stopping Criterion, Markov Chain, Variance

1. Introduction

Genetic Algorithms (GAs) are stochastic search methods based on the principles of natural genetic systems. They perform a multidimensional search in providing an optimal solution for evaluation (fitness) function of an optimization problem. Unlike the conventional search methods, GAs deal simultaneously with multiple solutions and use only the fitness function values. Population members are represented by strings, corresponding to chromosomes. Search starts with a population of randomly selected strings, and, from these, the next generation is created by using genetic operators. At each

[*]Address for correspondence: Center for Soft Computing Research, Indian Statistical Institute, Kolkata 700108, India

iteration individual strings are evaluated with respect to a performance criteria and assigned a fitness value. Strings are randomly selected using these fitness values to either survive or to mate to produce offspring for the next generation. All such strings are subject to mutation. GAs are theoretically and 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, machine learning etc. [6, 11, 16, 15, 17, 9, 12, 10].

Among the many global search methods available, GA has been considered to be a viable and promising option for exploration. This evolutionary technique is population oriented, successive populations of feasible solutions are generated in stochastic manner following laws of natural selection. In this approach, multiple stochastic trajectories approach simultaneously towards one or more regions of the search space providing important clues about the global structure of the function. Various theoretical aspects of genetic algorithms have been studied in the literature. Researchers have tried to establish the theoretical basis of the use of simple (yet difficult to model) operations. Attempts have been made to find the GAs amazing search ability by analyzing the evolution of strings generated by the crossover and mutation operations.

Genetic algorithms have been successfully modeled as Markov chains [9, 2, 3, 18, 14, 21, 20]. Vose [21], and Davis and Principe [3] have not preserved the knowledge of the previous best in their model. Bhandari et. al. [2], Rudolph [18] and Suzuki [20] preserved the knowledge of the previous best in their model and proved the convergence of GAs to the optimal string. In [13], Murthy et. al. have tried to provide the optimal stopping time for a GA in terms of ϵ-optimal stopping time. They have provided an estimate for the number of iterations that a GA has to run to obtain an ϵ-optimal global solution.

All of the steps of a GA are well defined except the stopping criterion. So far the practitioners use stopping criteria based on time or fitness value. Time based stopping criteria are mainly of two kinds. The popular one is to decide upfront the number of iterations to be executed. Another is based on execution time of the algorithm. Algorithm is run for a predetermined period and gets the result. Though these criteria are very simple to implement, determining the time is again a challenge. In the first, the process is executed for a fixed number of iterations and the best string, obtained so far, is taken to be the optimal one. While in the other, the algorithm is terminated if no further improvement in the fitness value for the best string is observed for a fixed number of iterations. However, it is not clear how to fix the number of iterations required for the execution of a GA.

In this article, we propose a new stopping criterion based on the variance of the best fitness values obtained over the generations. The proposed criterion uses only the fitness function values and takes into account the inherent properties of the objective function. User does not need to study the characteristics of the objective function and the genetic parameters to be used in the algorithm. The criterion based on the variance of the fitness function values obtained over the generations only needs a sufficiently small values of the bound. It has been shown theoretically that the variance tends to zero when the number of generations tends to infinity with probability of obtaining the global optimal solution tends to unity.

The basic principles of genetic algorithms and a description of GAs with elitist model (EGAs) are provided in the next section. The mathematical modeling of EGAs and the issues of convergence are discussed in Sections 3. Sections 4 and 5 deal with the optimal stopping time of EGAs. Section 6 presents experimental results demonstrating the effectiveness of the proposed criterion for a number of objective functions. A detailed comparative study among different stopping criteria is provided in Section 7. Section 8 concludes the article.

2. Basic Principles of Genetic Algorithms

We describe the basic principle of GAs in this section, 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^* such that

$$f(x^*) \geq f(x);\ \forall\, x \in D.$$

Note that D is a discrete domain since it is finite.

While solving an optimization problem using GAs, each solution is coded as a string (called ”chromosome”) of finite length (say, L). Each string or chromosome is considered as an individual. A collection of M (finite) such individuals is called a population. GAs start with a randomly generated population. In each iteration, a new population of 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.

To use Genetic algorithms in searching the global optimal solution, the very first step is to define a mechanism to represent the solutions in a chromosomal form. The pioneering work of Holland proposed to represent a solution as a string of length L over a finite set of alphabet $\mathcal{A}$. Each string S corresponds to a value $x \in D$. The GA with $\mathcal{A} = \{0, 1\}$ is termed as binary coded genetic algorithm (BCGA) or simple genetic algorithm (SGA). The string representation limits the algorithm to search a finite (though users can achieve their required approximation by increasing the string length) domain and provides the best solution among the 2^L possible options. To take into account the continuous domain, real valued strings are considered as the chromosomal representation by suitably manipulating the genetic operators and is termed as real coded genetic algorithm (RCGA). However, it is again quite difficult to consider all the real values considering the limitation of the computer in storing irrational values. Henceforth, throughout this article, we shall take $\mathcal{A} = \{0, 1\}$ as this paper deals with the SGAs and can be easily extended to GAs defined over a finite set of Alphabet or over RCGAs.

Generally a random sample of size M is drawn from 2^L possible strings to generate an initial population. GAs leverage a population of solutions to generate a new population with the expectation that the new population will provide better solution in terms of the fitness values.

In every iteration, we evaluate each chromosome of the population using fitness function fit. Evaluation or fitness function fit for a string S is equivalent to the function f:

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

where, S corresponds to x. Without loss of generality, let us assume that $fit(S) > 0$ for all S in ***S*** where, ***S*** is the set of all possible strings.

Selection is a process in which 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.

Crossover exchanges information between two potential strings and generates two offspring for the next population. Mutation is an occasional random alteration of a character. Mutation introduces some extra variability into the population. Though it is performed usually with very low probability $q(>0)$, it has an important role in the exploration process. Every character in each chromosome (generated after crossover) has an equal chance to undergo mutation. Note that, any string can be generated from any given string by mutation operation [13]. The mutation probability q is taken to be in the range of $(0, 0.5]$. It may be due to the fact that, intuitively, the probability of mutating i bit positions is more than that of mutating $i+1$ bit positions, i.e.,

$$q^i(1-q)^{L-i} > q^{i+1}(1-q)^{L-i-1} \quad \forall i = 0, 1, 2, ..., L-1$$

which results in $q \leq 0.5$. Hence the minimum probability of obtaining any string from any given string is q^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 preserved either (i) in a separate location outside the population or (ii) within the population; in that way the algorithm would report the best value found, among all possible coded solutions obtained during the whole process. GAs with this strategy of retaining the knowledge of the best string obtained so far as genetic algorithms with *elitist model* or *EGAs*. 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 M is finite. This paper deals with the GAs with the elitist model (EGA) of selection of De Jong [4], where the best string obtained in the previous iteration is copied into the current population if the fitness function values of all strings are less than the previous best.

Note that the values for the parameters L, M, p and q have to be chosen 'properly' before performing those operations. The population size M is taken as an even integer so that strings can be paired for crossover. The probability (p) of performing crossover operation is taken to be any value between 0.0 and 1.0. Usually in GAs, p is assumed to be a value in the interval $[0.25, 1]$ [6]. The mutation probability q is taken to be very low $[0.001, 0.01]$ [6], however, it can be taken in the interval $(0, 0.5]$. Mutation plays an important role in the convergence of GAs to the global optimal solution. The following section presents the approach Bhandari et. al. [2] provided to prove the convergence of GAs as that is the fundamental building block in proposing the variance of the best fitness value obtained so far as the stopping criterion of a GA.

3. Convergence of Genetic Algorithms

Various theoretical aspects of genetic algorithms have been studied in the literature. Researchers have tried to establish the theoretical basis of the use of simple (yet difficult to model) operations. Attempts have been made to find the GAs amazing search ability by analyzing the evolution of strings generated by the crossover and mutation operations. Genetic algorithms have been successfully modeled as Markov chains [2, 3, 18, 14, 21, 20]. Vose [21], and Davis and Principe [3] have not preserved the knowledge of the previous best in their model. Bhandari et. al. [2], Rudolph [18] and Suzuki [20] preserved the knowledge of the previous best in their models and proved the convergence of GAs to the optimal string. An extensive theoretical study regarding the dynamics of evolutionary algorithms may be found in [9].

Genetic algorithms search over a space $\boldsymbol{S}$ of 2^L strings and eventually provide the best with respect to the fitness function fit. The strings can be classified into a set of s classes depending on their fitness function values. The classes are defined as

$$\boldsymbol{S}_i = \{S : fit(S) = F_i\}$$

where F_i denotes the ith highest fitness function value. Thus, $F_1 > F_2 > \cdots > F_s$. Let us also assume without loss of generality that $F_s > 0$.

A population Q is a multi-set of M strings of length L generated over a finite alphabet A and is defined as follows:

$$\begin{aligned} Q &= \{S_1, S_1, \cdots, (r_1\, times), S_2, S_2, \cdots, (r_2\, times), \cdots, S_m, S_m, \cdots, (r_m\, times) \\ &\quad \text{where, } S_i \in \boldsymbol{S};\ S_{i_1} \neq S_{i_2}\ \forall i_1 \neq i_2 \text{ and } r_i \geq 1\ for i = 1, 2, \cdots, m; \textstyle\sum_{i=1}^{m} = M\}. \end{aligned}$$

Let $\boldsymbol{Q}$ denote the set of all populations of size M. The number of populations or states in the Markov chain is finite.

The fitness function value $fit(Q)$ of a population is defined as $fit(Q) = \max_{S \in \boldsymbol{Q}} fit(S)$. Then the populations are partitioned into s sets. $E_i = \{Q : Q \in \boldsymbol{Q}$ and $fit(Q) = F_i\}$ is a set of populations having the same fitness function value F_i. In an iteration, the genetic operators (selection, crossover and mutation) create a population $Q_{kl} \in E_k; l = 1, 2, \cdots, e_k$ and $k = 1, 2, \cdots, s$; from some $Q_{ij} \in E_i$ where e_k is the number of elements in E_k. The generation of a population Q_{kl} from Q_{ij} is considered as a transition from Q_{ij} to Q_{kl} and let $p_{ij.kl}$ denotes this transition probability. Then the probability of transition from Q_{ij} to any population in E_k can be calculated as

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

For all $j = 1, 2, \cdots, e_i$ and $i = 1, 2, \cdots, s$ one obtains

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

by construction. This means that once GAs reach a population $Q \in E_k$ they will always be in some population $Q' \in E_k$ for $k \leq i$. In particular, once GAs reach a population $Q \in E_1$ they will never go out of E_1.

Let $p_{ij.kl}^{(n)}$ be the probability that GA results in Q_{kl} at the nth step given that the initial state is Q_{ij}. Let $p_{ij.k}^{(n)}$ denote the probability of reaching one of the populations in E_k from Q_{ij} at the nth step. Then $p_{ij.k}^{(n)} = \sum_{l=1}^{e_k} p_{ij.kl}^{(n)}$.

To show the eventual convergence of a GA with elitist model to a global optimal solution the following theorem has been proved in [2] and is not presented here.

Theorem 1. For an EGA with the probability of mutation $q \in [0, \frac{1}{2}]$,

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

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

4. Stopping Criteria

Proof of convergence of an algorithm to an optimal solution is very important as it assures the optimal solution in infinite iterations. When an algorithm does not assure or guarantee the optimal solution even after running for infinite iterations, it's utility is in doubt. Once the convergence of an algorithm is assured, the focus turns into the exploration of stopping time or stopping criterion of the algorithm. Today, the biggest challenge in the implementation of GAs is to decide when to stop the algorithm keeping in mind that there is no a-prior information regarding the objective function. Attempts have been made to provide stopping criteria of a GA based on time the algorithm is being executed and obtained objective function values or their distribution [11, 8, 12]. Time based stopping criteria are mainly of two kinds. The popular one is to decide upfront the number of iterations to be executed. Another is based on execution time of the algorithm. Algorithm is run for a predetermined period and gets the result. Though these criteria are very simple to implement but determining the time is again a challenge. This would require a good knowledge about the global optimal solution, which is not always available. In the first, the process is executed for a fixed number of iterations and the best string, obtained so far, is taken to be the optimal one. While in the other, the algorithm is terminated if no further improvement in the fitness value for the best string is observed for a fixed number of iterations. Though these criteria are easy to implement, they do not guarantee the convergence of the GAs to the global optimal solution as they are terminated after a finite number of iterations.

The criteria based on objective function values use the underlying fitness function values to calculate auxiliary values as a measure of the state of the convergence of the GA. Subsequently, the running mean, standard deviation of the population under consideration, Best Worst, Phi, Kappa were defined as a convergence measure. In [8], Jain et. al. have tried to provide a cluster based stopping criteria called ClusTerm. This concept takes into account the information about the objective values as well as the spatial distribution of individuals in the search space in order to terminate a GA.

Safe et. al. [19] presented a critical analysis of various aspects associated with the specification of termination conditions for simple genetic algorithms. The study, which is based on the use of Markov chains, identifies the main difficulties that arise when one wishes to set meaningful upper bounds for the number of iterations required to guarantee the convergence of such algorithms with a given confidence level. Greenhalgh et. al. [7] have discussed convergence properties based on the effect of mutation and also obtained an upper bound for the number of iterations necessary to ensure the convergence. In [1], authors have tried to provide a stopping criteria together with an estimation for the stopping time. In [13], Murthy et. al have provided ϵ-optimal stopping criteria for GAs with elitist model and also derived the ϵ-optimal stopping time. Pessimistic and optimistic stopping times were derived with respect to the mutation probability. However, it is to be noted that with existing operations, whatever value for the number of iterations N is decided, there is always a positive probability of not obtaining the global optimal solution at that stage and is stated below as a lemma.

Lemma 1. With existing operations, whatever value for the number of iterations N is decided, there is always a positive probability of not obtaining an optimal solution.

Proof:
In general, the number of solutions in the solution space Ω is much higher compare to the number of optimal solutions. Moreover, the population size considered in the implementation of genetic algorithms

is very low compare to that of solution space. Therefore, it is clear that the probability of starting with a population (Q) containing no optimal solution is positive. Let $P^{(1)}_{Q.Q}$ be the probability of obtaining the same population in one iteration. It is obvious that $P^{(1)}_{Q.Q} > 0$ as the probability of mutation δ is assumed to be $<\ 1$. Now,

$$\begin{aligned} P^{(2)}_{Q.Q} &= \sum_{Q' \neq Q} P^{(1)}_{Q.Q'}.P^{(1)}_{Q'.Q} + P^{(1)}_{Q.Q}.P^{(1)}_{Q.Q} \\ &> P^{(1)}_{Q.Q}.P^{(1)}_{Q.Q} \\ &> 0. \end{aligned}$$

Similarly, $P^{(n)}_{Q.Q}$, the probability remain in Q in n iteration, is > 0. □

The theoretical study brings out a number of limitations in defining a stopping criterion. Some desirable properties of a "good" stopping criterion are given below.

- Easy to implement.
- Able to provide stopping time automatically for any fitness function.
- Should lead to 'good' / 'satisfactory' result.
- Total number of strings searched should not exceed 2^L.

However, the fundamental task of a stopping criterion is to provide a guideline in terminating the algorithm so that the solution obtained at that time and the optimal solution are close to each other. In other words, the stopping criterion provides the user a guideline in stopping the algorithm with an acceptable solution close to the optimal solution. Mathematically, the closeness may be judged in various ways. One way is to show that the probability of not reaching optimal at that time is less than a small predefined quantity. Another way is to measure the distance between the optimal and the current solution, and show that the distance is less than a small predefined quantity. If stopping criterion is not mathematically (i.e., heuristically) defined, the amount of error (the value of probability in first case and distance in the second case) in accepting the solution would not be known.

In this context, a new stopping criterion based on the variance of the best solutions obtained up to the iteration in hand is defined here. This is easily implementable and does not need any auxiliary information other than the fitness function values obtained so far. The theoretical study given below reveals its properties and strengths in searching for the global optimal solution.

5. Proposed Stopping Criterion

Let a_i be the best fitness function value obtained at the end of ith iteration of an EGA. Then, $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_n \leq \cdots \leq F_1$, as F_1 is the global optimal value of the fitness function (defined in section 3). Let $\bar{a_n} = \frac{1}{n}\sum_{i=1}^{n} a_i$ be the average of the a_i's and $\bar{a^2_n} = \frac{1}{n}\sum_{i=1}^{n} a_i^2$ be the average of the a_i^2's up to the nth iteration, then variance of the best fitness values obtained up to the nth iteration, defined by b_n, is

$$b_n = \frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a_n})^2 = \frac{1}{n}\sum_{i=1}^{n} a_i^2 - \bar{a_n}^2 = \bar{a^2_n} - \bar{a_n}^2$$

b_n can be used as a stopping criterion for a GA. A GA is stopped or terminated after N iterations when $b_N < \epsilon$, where $\epsilon(> 0)$ is a user defined small quantity.

Given below are the basic steps of the genetic algorithm with elitist model where variance of the best solutions obtained in the generations is considered as a stopping criterion.

1. A population of random solutions is created and an ϵ is defined.
2. Each solution is evaluated on the basis of the fitness function.
3. Store the best solution if it is better than previous best.
4. Calculate the variance of the best solutions obtained so far.
5. If the variance is greater than the predefined value (ϵ), go to the next step, else stop the algorithm
6. A new generation of solutions is created from the old generation using selection, crossover and mutation.
7. Steps 2 to 6 above are repeated until the condition in step 5 is satisfied.

Now, we will theoretically establish that when the number of generations tends to infinity, the probability of obtaining the global optimal solution tends to 1, and the variance of the best fit solutions obtained in the generations approaches to 0.

In Section 3, we have discussed the convergence of GAs with elitist model. In the convergence theorem, we have seen that

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

The convergence theorem in turn implies that the probability of obtaining a global optimal solution (F_1)is 1 as number of iterations goes to infinity can be stated as the following lemma.

Lemma 2: For each $\epsilon_1 > 0$, $\lim_{n\to\infty} Prob(|a_n - F_1| > \epsilon_1) = 0$. In other words, for each $\epsilon_0 > 0$ and $\epsilon_1 > 0$, there exists N_0 such that for $n > N_0$,

$$\begin{array}{rlcl} & 1 - Prob(|a_n - F_1| \leq \epsilon_1) & < & \epsilon_0 \text{ or} \\ \Rightarrow & Prob(|a_n - F_1| \leq \epsilon_1) & > & 1 - \epsilon_0 \text{ for } n > N_0 \end{array} \tag{1}$$

Proof: Trivial.

With the help of the above lemma, we shall now show that the variance of the best solutions obtained in the generations approaches to 0 when number of iterations tends to ∞.

Theorem 2: $Prob(\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a_n})^2 \leq \epsilon) \to 1$ as $n \to \infty$ for each $\epsilon > 0$.

Proof:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{a_n})^2 &= \frac{1}{n}\sum_{i=1}^{n}[(a_i-F_1)-(\bar{a_n}-F_1)]^2\\ &= \frac{1}{n}\sum_{i=1}^{n}(a_i-F_1)^2-(\bar{a_n}-F_1)^2\\ &\leq \frac{1}{n}\sum_{i=1}^{n}(a_i-F_1)^2\end{aligned} \tag{2}$$

Now for $n > N_0$,

$$\frac{1}{n}\sum_{i=1}^{n}(a_i-F_1)^2=\frac{1}{n}\sum_{i=1}^{N_0}(a_i-F_1)^2+\frac{1}{n}\sum_{i=N_0+1}^{n}(a_i-F_1)^2$$

Since F_s is the minimum value of the function $f(x)$ (defined in section 3), we have,

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{N_0}(a_i-F_1)^2 &\leq \frac{1}{n}\sum_{i=1}^{N_0}(F_s-F_1)^2 \text{ (as } F_s \leq a_i \leq F_1\ \forall i)\\ &= \tfrac{N_0}{n}(F_s-F_1)^2\end{aligned} \tag{3}$$

One can always find an N_1 ($> N_0$) such that for each ϵ_2 (> 0),

$$\frac{N_0}{N_1}(F_s-F_1)^2<\epsilon_2 \tag{4}$$

Therefore, for $n > N_1 > N_0$

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{N_0}(a_i-F_1)^2 &\leq \tfrac{N_0}{n}(F_s-F_1)^2\\ &\leq \tfrac{N_0}{N_1}(F_s-F_1)^2\\ &\leq \epsilon_2 \quad \text{from (4)}\end{aligned} \tag{5}$$

As $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_i \leq a_{i+1} \leq \cdots \leq F_1$,

$$\begin{aligned}\frac{1}{n}\sum_{i=N_0+1}^{n}(a_i-F_1)^2 &\leq \frac{1}{n}\sum_{i=N_0+1}^{n}(a_{N_0+1}-F_1)^2\\ &= \tfrac{n-N_0-1}{n}(a_{N_0+1}-F_1)^2\\ &\leq (a_{N_0+1}-F_1)^2\end{aligned} \tag{6}$$

From (1), we have for $n > N_0$,

$$Prob(|a_n-F_1|\leq\epsilon_1)>1-\epsilon_0$$

Therefore,

$$\begin{aligned} & Prob((a_{N_0+1}-F_1)^2 \leq \epsilon_1^2) > 1-\epsilon_0\\ \Rightarrow\ & Prob((a_{N_0+1}-F_1)^2 \leq \epsilon_1) > 1-\epsilon_0, \text{ as } \epsilon_1 << 1.\end{aligned} \tag{7}$$

Now, for each $\epsilon = \epsilon_1 + \epsilon_2$,

$$\begin{aligned}
& Prob(\frac{1}{n}\sum_{i=1}^{n}(a_i - F_1)^2 \leq \epsilon) \\
= \ & Prob(\frac{1}{n}\sum_{i=1}^{N_0}(a_i - F_1)^2 + \frac{1}{n}\sum_{i=N_0+1}^{n}(a_i - F_1)^2 \leq \epsilon) \\
\geq \ & Prob(\epsilon_2 + \frac{1}{n}\sum_{i=N_0+1}^{n}(a_i - F_1)^2 \leq \epsilon) \ \text{(from 5)} \\
= \ & Prob(\frac{1}{n}\sum_{i=N_0+1}^{n}(a_i - F_1)^2 \leq \epsilon - \epsilon_2) \\
> \ & 1 - \epsilon_0, \ \text{(from 7), where } \epsilon_1 = \epsilon - \epsilon_2.
\end{aligned} \tag{8}$$

Therefore, we can conclude that for each $\epsilon_0 > 0$, there exists an N_1 such that for $n > N_1$

$$Prob(\frac{1}{n}\sum_{i=1}^{n}(a_i - F_1)^2 < \epsilon) > 1 - \epsilon_0$$

In other words, $Prob(\frac{1}{n}\sum_{i=1}^{n}(a_i - F_1)^2 \leq \epsilon) \to 1$ as $n \to \infty$ for each $\epsilon > 0$.

This completes the proof of theorem 2. □

The following remarks can be made regarding the proposed algorithm with variance as stopping criterion.

Remarks:

1. The variance is calculated from the fitness values obtained over the generations, which implicitly takes into account the characteristics of the objective function.

2. ϵ signifies a measure of error, the difference between the fitness value of the best solution obtained so far and the global optimal solution.

5.1. Implementation Details

Some of the salient features of the stopping criterion are discussed below regarding its implementation.

- The user needs to choose only the bound for variance for implementing this criterion. Naturally, less the value of the bound, more is the chance of obtaining a solution close to global optima. A user can decide the value of ϵ based on the accuracy required.

- Due to the randomness involved in the algorithm, it may so happen that there may not be any change (improvement) in the best fitness value over a number of consecutive iterations which will result in 0 variance. It is important to select a significant number of iterations from which the fitness values will be considered in calculating the variance so that the algorithm gets enough opportunity to yield improved (better) solution.

- Variance can be iteratively calculated. Variance b_{n+1} at the end of $(n+1)$th iteration can be calculated as follows:

$$b_{n+1} = \frac{1}{n+1}\sum_{i=1}^{n+1}(a_i - \bar{a_{n+1}})^2 = \frac{1}{n+1}\sum_{i=1}^{n+1} a_i^2 - \bar{a_{n+1}}^2$$

  or

$$b_{n+1} = \frac{1}{n+1}((n\bar{a_n^2} + a_{n+1}^2) - (n\bar{a_n} + a_{n+1})^2) \tag{9}$$

  This indicates that only the average of the fitness function values and their squares of the previous n iterations are required to evaluate the variance for the $n+1$th generation. This iterative feature of variance calculation indeed makes the algorithm easier to implement.

- The proposed variance based criterion is that it is not scale invariant. That means it is sensitive to transformations of the fitness function. The algorithm may need different number of iterations for $f(x)$ and $g(x) = k * f(x)$, where k is a constant. Sometimes, scale invariance is a desirable property but not always. There are several measures that are not scale invariant, e.g., mean, variance, co-variance, moments. However, one can easily avoid the impact of the scaling effect by a simple transformation of the fitness function. One such transformation is given in equation (10).

$$g(x) = \frac{f(x)}{f_{max}^{(1)}} \tag{10}$$

  where, $f_{max}^{(1)}$ is the maximum value of the fitness function obtained in the first iteration.

  Let us now try to understand the impact of the above mentioned transformation in the selection of the value of ϵ. We have,

$$b_n = \frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a_n})^2 \tag{11}$$

  Now, let $b_n(g)$ be the variance of the best fitness values obtained up to the nth iteration for the function $g(x)$. Then, it is clear that

$$b_n(g) = \frac{1}{n}\frac{\sum_{i=1}^{n}(a_i - \bar{a_n})^2}{{f_{max}^{(1)}}^2} \tag{12}$$

  It is now obvious that the user who assumed ϵ_f as the value of ϵ for the function f can assume $\epsilon_f * {f_{max}^{(1)}}^2$ as the value of ϵ for the function g. This simply implies that the user has to adjust the value of ϵ for the applied transformation. It may be convenient for the user to select the value of ϵ after the transformation. Therefore, the user needs to know his/her desire of making the stopping criterion scale invariant based on the convenience for selecting the value of ϵ.

The experimental results are being discussed in the following section.

6. Experimental Results

The effectiveness of the proposed stopping criterion is demonstrated in searching for global optimal solutions of some complex functions of multiple variables. A number of typical objective functions (given below) are used in the experiment [8].

$$\begin{aligned} f_1(x) &= 6 + sin(x) \text{ when } 0 \le x \le 2\pi \\ &= 6 + 2sin(x) \text{ when } 2\pi < x \le 4\pi \\ &= 6 + 3sin(x) \text{ when } 4\pi < x \le 6\pi \\ &= 6 + 4sin(x) \text{ when } 6\pi < x \le 8\pi \\ &= 6 + 5sin(x) \text{ when } 8\pi < x \le 10\pi \\ &= 6 + sin(x) \text{ when } 10\pi < x \le 32 \end{aligned}$$

$$f_2(x) = \log(1 + \sum_{i=1}^{5} |[x_i]| + \prod_{i=1}^{5} |x_i|), \text{ where } [x] \text{ is the integral part of } x$$

$$f_3(x) = \frac{1}{1 + \sum_{i=1}^{5} [x_i]^2}, \text{ where } [x] \text{ is the integral part of } x$$

$$f_4(x) = \frac{20}{11 + \sum_{i=1}^{5} -[x_i] * sin(\sqrt{|[x_i]|})}, \text{ where } [x] \text{ is the largest integer } \le x$$

The pictorial presentations of the functions are shown in fig 1 (a-d). f_1 is a univariate function while the remaining 3 are multi-variate (number of variable is considered as 5 here). Functions f_2 and f_3 are multimodal with symmetrical distributed plateaus of identical size and having multiple global maxima. f_1 and f_4 are unimodal with spatially distant local maxima and single global maxima. Different search spaces are considered for different functions to exploit the typical features of the functions.

As mentioned earlier, there may be no change in the fitness function value for a number of consecutive iterations. Therefore, the variance would become 0 and may result in premature termination of the algorithm. The user should consider a significant number of iterations in calculating the variance. In our experiment the minimum number of iterations considered to generate variance are 50 for f_1 and 200 for other functions.

The genetic parameters used in the execution of the algorithm are as follows:

Population size	=	10 for f_1 and 50 for others
String length	=	20 for f_1 and 100 for others
Crossover probability	=	0.8
Mutation probability	=	varying from 0.2 to 0.45

To obtain statistical significant results, one test run comprises 100 runs for a particular ϵ value for each function. Different seeds are being supplied to bring in the randomness in generating initial populations

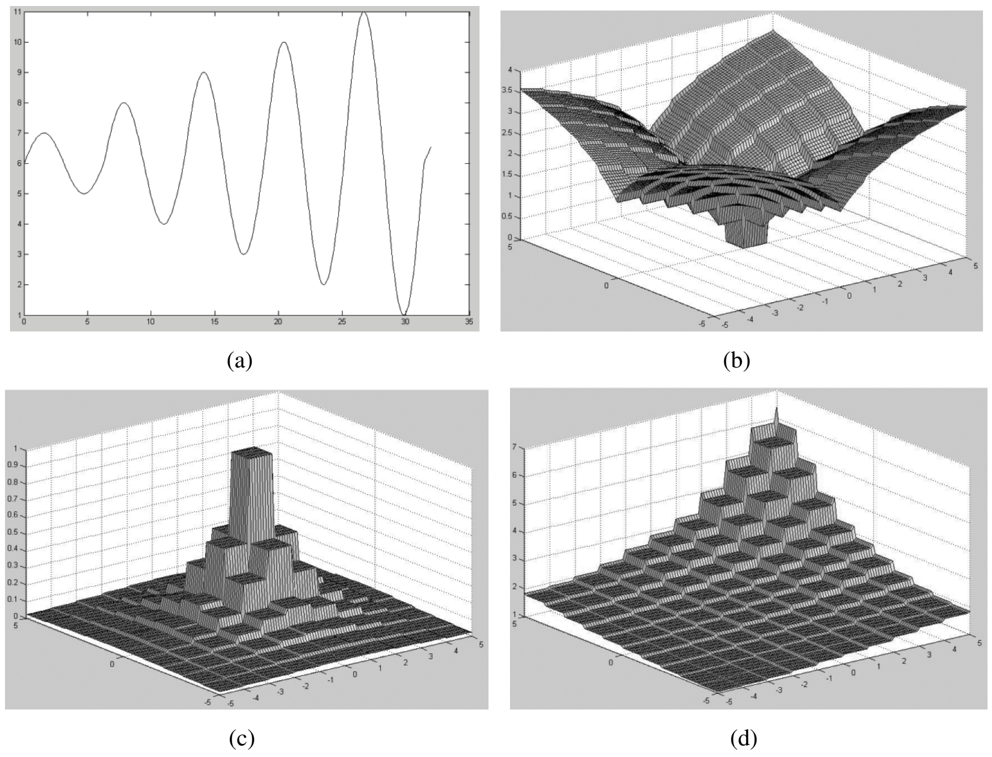

Figure 1. Pictorial presentation of the functions

and performing other genetic operations. Considering the importance of mutation in the convergence process of the GAs, the mutation probability is made variable. It is considered as high as 0.45 in the initial iterations and being monotonically reduced to 0.1 towards the final iterations. Fig. 2 depicts the trend of the variance with the iteration number. It is clearly non-decreasing. As the algorithm explores better solution with higher fitness value the variance increases.

Table 1 depicts the average number of iterations required in order to converge the algorithm for a given ϵ. The results shows that for a low value of ϵ, the algorithm produces satisfactory performance for all the functions. In fact, the algorithm produces global optimal solution in most of the cases for $\epsilon = 10^{-5}$. Note that the number of iterations to attain the given bound differs for different functions depending on the characteristics of the function. Also the percentage of convergence to the global optimum solution for f_3 is much higher whereas it is lower for f_4 (in fact, with values $> 10^{-4}$ of ϵ, no run could produce the global optimal solution). This is due to the fact that the presence of multiple global optima of f_3 results in faster convergence while the single optima of f_4 is hard to attain. This clearly demonstrates the effectiveness of the criterion to take into account the inherent properties of the objective unction. In some cases though the stopping criterion is satisfied, the algorithm does not converge to the global optimal value of the objective function. This is in line with the fact that GAs do not guarantee the global optimal solution in a finite number of iterations. However, with the reduction in ϵ value the chance of obtaining the global optimal solution increases.

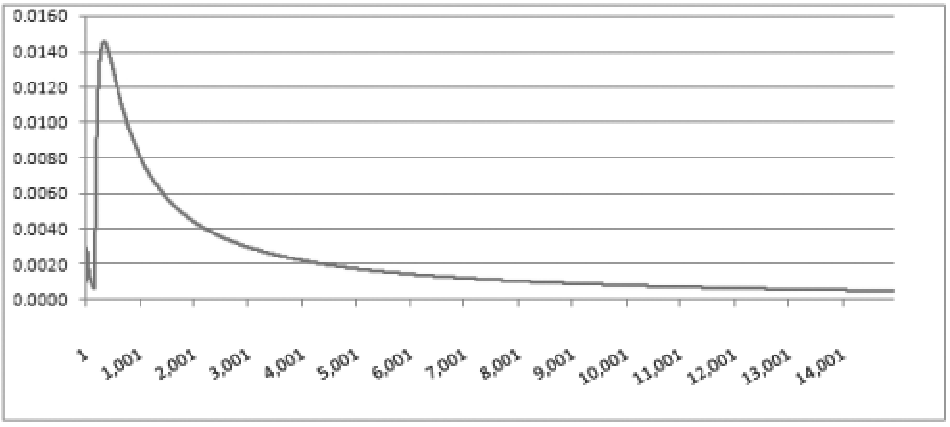

Figure 2. Change in variance with iteration

7. Differences with Other Stopping Criteria

Three widely used stopping criteria are (i) the variance of the fitness evolved in the whole population is less than a predefined small quantity (ϵ_p, say), (ii) number of iterations is greater than or equal to a fixed number (N), decided a-priori, and (iii) no improvement in the best fitness value through a fixed number of iterations (say K). Let us refer them as POP-Var, AN-it and K-it respectively in the successive discussion.

7.1. Pop-var

In this framework, the algorithm stops when the variance of fitness values of all the strings in the current population is less than a predefined threshold (say, ϵ_p). Usually, ϵ_p is taken to be a small value, close to 0. It is primarily based on the assumption that after significantly many iterations the fitness values of the strings present in the population are all close to each other, thereby making the variance of the fitness values close to 0. In general, this assumption is not true due to the following reasons: (i) Usually in elitist model only the best string is preserved, (ii) any population containing an optimal string is sufficient for the convergence of the algorithm and (iii) there is a positive probability of obtaining a population after infinitely many iterations with exactly one optimal string and others are being not optimal. This is further illustrated using an example taken from [2].

In this example, a maximization problem is considered in the domain $D = \{0, 1, 2, 3\}$. We have considered $M = 2,\ L = 2$ and $\mathcal{A} = \{0, 1\}$. 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

$$fit(s_1) = 1 fit(s_2) = 4 fit(s_3) = 2 \text{ and} fit(s_4) = 3.$$

The strings can be classified into four classes which are $\mathcal{S}_1 = \{s_2\}$, $\mathcal{S}_2 = \{s_4\}$, $\mathcal{S}_3 = \{s_3\}$ and $\mathcal{S}_4 = \{s_1\}$.

Function	ϵ	Average number of Iterations	% of cases when global solution is reached	Number of strings in search space	Number of strings searched
f_1	10^{-2}	208.01	29		$2,080$
	10^{-3}	541.04	43	$2^{20} \approx 10^6$	$5,410$
	10^{-4}	4421.97	62		$44,219$
	10^{-5}	28207.21	78		$282,072$
f_2	10^{-2}	938.36	12		$46,918$
	10^{-3}	16625.88	23	$2^{100} \approx 10^{30}$	$831,294$
	10^{-4}	203736.02	37		$10,186,801$
	10^{-5}	1260294.83	51		$63,014,741$
f_3	10^{-2}	821.04	89		$41,052$
	10^{-3}	5298.25	100	$2^{100} \approx 10^{30}$	$264,912$
	10^{-4}	36175	100		$1,808,750$
	10^{-5}	99980.98	100		$4,999,049$
f_4	10^{-2}	235.78	0		$11,789$
	10^{-3}	1289.42	0	$2^{100} \approx 10^{30}$	$64,471$
	10^{-4}	14392.51	0		$719,625$
	10^{-5}	92406.10	20		$4,620,305$

Table 1. Average iterations for various ϵ

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.

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

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

$$\mathcal{P}^{(1)} = \begin{pmatrix} 0.960596 & 0.009999 & 0.009803 & 0.019602 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.318435 & 0.667011 & 0.008056 & 0.006498 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.426975 & 0.228811 & 0.331134 & 0.013080 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.617890 & 0.009608 & 0.010230 & 0.362272 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.000098 & 0.019406 & 0.000196 & 0.000002 & 0.960696 & 0.019506 & 0.000196 & 0 & 0 & 0 \\ 0.000036 & 0.007081 & 0.004760 & 0.000048 & 0.350488 & 0.632812 & 0.004775 & 0 & 0 & 0 \\ 0.000098 & 0.014555 & 0.180271 & 0.004853 & 0.540372 & 0.019506 & 0.240345 & 0 & 0 & 0 \\ 0 & 0.000002 & 0.000196 & 0.000002 & 0.000098 & 0.019406 & 0.000196 & 0.960596 & 0.019504 & 0 \\ 0.000011 & 0.000045 & 0.004422 & 0.002244 & 0.000044 & 0.008712 & 0.004422 & 0.431255 & 0.548815 & 0 \\ 0.000098 & 0.000002 & 0.000196 & 0.019406 & 0 & 0.000002 & 0.000196 & 0.000098 & 0.019406 & 0.960596 \end{pmatrix}$$

$$\mathcal{P}^{(1024)} = \begin{pmatrix} 0.918654 & 0.038293 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918652 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918652 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918651 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918646 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918648 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918649 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918628 & 0.038291 & 0.014366 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918632 & 0.038292 & 0.014367 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.918631 & 0.038292 & 0.014366 & 0.028922 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The figures in the above matrices are given upto 6 decimal places. It can be easily seen from the matrix $\mathcal{P}^{(1024)}$ that $p_{i.j}^{(1024)}$ are nearly zero for all $j \geq 5$ and for all $i = 1, 2, \cdots, 10$. It can also be seen that all the rows are almost identical in $\mathcal{P}^{(1024)}$. From the literature on Markov chains [5], it is necessary that all the rows of $\mathcal{P}^{(n)}$ are 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.

The POP-Var criterion uses the variance of the population as the basis in the termination of the algorithm. This criterion will be effective when the population becomes homogeneous (having the same or almost same fitness values of all the strings). For the given example, the algorithm obtains Q_1, Q_5, Q_8 or Q_{10} (Q_1 being the most desired population) so that the algorithm converges to the global optimal solution satisfying the POP-Var criterion. It is clear from $\mathcal{P}^{(1024)}$ that the probability of obtaining Q_1 after 1024 iterations is maximum but there is a significant chance in obtaining Q_2, Q_3 or Q_4 after those many iterations. The elements of second, third and fourth columns of $\mathcal{P}^{(n)}$ are positive (> 0) even after $n > 1024$. Therefore, the algorithm does not guarantee the maintenance of the same population over the iterations. Moreover, there is a high chance of premature termination.

7.2. AN-it

In this stopping criterion, the number of iterations to be executed is decided a-priori. Let us denote the fixed value by N. It has been proved that as the number of iterations $n \to \infty$, we shall obtain a population containing an optimal string with probability 1. Note that any value N, that is fixed to a finite number, whereas the bound on the number of iterations, i.e., ∞, is infinite. One would like to fix, in general, a value so that the fixed value is closed to the limiting value. Here, we need to make the number of iterations to be infinite to obtain the optimum value. Any finite value, which is fixed, will have infinite difference with ∞. Thus, though higher the value of N, the better would be the confidence on the obtained result, the difference between ∞ and the fixed value will remain ∞. Ideally, one would like to fix a value so that the difference between optimal and the fixed value can be measured and is small. These two properties are upheld by the proposed stopping criterion.

7.3. K-it

In this criterion, if there is no change in the best fitness value for K consecutive iterations, the algorithm is terminated. The value of K is to be fixed by the user. So the basic assumption, the user is making here is that it is impossible to obtain a better string after this K consecutive iterations. Actually, following *lemma* 1, one can show that there is always a positive probability of obtaining K consecutive equal sub-optimal solutions, whatever may be the value of K, where K is finite. However, if $K \to \infty$, for an elitist model, the probability of reaching an optimal solution will also tend to 1. As in AN-it, the upper bound of K is ∞ and we can not get a finite difference between K and ∞.

Note that starting with a sub-optimal population Q (Q does not contain any optimal string), there is always a positive probability to remain in the same sub-optimal population Q after K iterations. This probability will be higher if Q represents a sub-optimal population as well as a local optimal population (a population containing a local optimal solution). In this situation, one needs to continue the algorithm further to make it go out of the local optimal population. Additionally, the same value of K need not hold good for every local optimal population. One needs to device a technique where, the algorithm takes into account these situation automatically. The proposed criterion partially does this job.

7.4. Proposed Criterion

The proposed criterion fixes a bound (ϵ) on the variance of the best fitness values obtained through a number of iterations, and the algorithm stops when the variance is less than the bound. It has been shown in this article that the variance goes to 0 with probability 1 as number of iterations goes to ∞. Thus ϵ, which is equal to $\epsilon - 0$, can be viewed as the difference between the optimal and the best solution found so far. This measurement of the difference could not be done with AN-it or K-it. Secondly, by taking ϵ close to 0, we can improve upon the best fitness value.

Suppose, the genetic algorithm got stuck at a local optimal string/population. In this scenario, it is highly probable that for many consecutive iterations the best fitness value will remain the same. It may be noted that in order to reach the local optimal population, the algorithm would have gone through some iterations previously, because of which there would be some fitness values which are less than that of the current best fitness value. This enables the variance to be positive (> 0). By making the value of ϵ to be very small, the proposed criterion allows the algorithm to execute more number of iterations compare to the number of iterations allowed by K-it; thereby increasing the probability of going out of the local optimal population with the proposed criterion.

Let us consider two examples,

$$\begin{array}{lrcll} \text{Example 1:} & fit(S) & = & 2 & \text{when, } S = S_1 \\ & & = & 1 & \text{otherwise} \end{array}$$

$$\begin{array}{lrcll} \text{Example 2:} & fit(S) & = & 3 & \text{when, } S = S_1 \\ & & = & 2 & \text{when, } S = S_2 \\ & & = & 1 & \text{otherwise} \end{array}$$

In case of Example 1, when the string length is quite large (say > 100), it is likely that the starting population will not contain the optimal string S_1 and the probability of exploring the optimal string in significant number of iterations will be very low. In this scenario, the chance of obtaining the string with same fitness value over a significant number of iterations is very high. As a consequence the variance of the best fitness values obtained over the iterations remain 0 and the proposed criterion may not be applicable. Same will be the performance of Pop-var, AN-it and K-it. Similar result may be observed for the second example also. Moreover, once the second best string S_2 is explored, since the variance for the proposed criterion will be positive (> 0), it will allow the algorithm to run for a longer period thereby will increase the chance of exploring the optimal string S_1 contrary to to AN-it and K-it.

Though the characteristics of POP-Var, AN-it and K-it seem to be similar to the proposed variance based criterion, there are quite a few differences as mentioned below.

1. The proposed criterion not only takes into account the information extracted in a fixed number (say, K) of iterations but also considers the fitness values observed prior to those K iterations.

2. one may note that both the iteration based criteria (AN-it and K-it), mentioned above, are based on the fact that the algorithm converges as the number of iteration tends to ∞, whereas the proposed criterion is based on the fact that the difference between the global optimal value of the function and the fitness function value tends to 0.

3. It is intuitive to assume the value of N or K for AN-it or K-it respectively depending on the length of the string length L. When L is low one considers a low value for N or K. On the other hand, for a larger L, one will consider a higher value for K or N. While in the case of proposed criterion, one does not need to change the value of ϵ.

4. In AN-it and K-it, one has to define the number at the start of the algorithm. While for the proposed algorithm, one does not need to define the number of iterations to be executed.

5. For the iteration based criteria, one decides the number of iterations keeping in mind that the algorithm will provide a satisfactory result after those many iterations. This is purely heuristic without considering the characteristics of the function. The inherent characteristics of the objective functions are automatically taken into account for a sufficiently small value of the bound for variance in the proposed criterion.

6. The proposed criterion clearly gives a finite measure regarding the closeness of the obtained solution to an optimum solution.

7. While POP-Var have used the information from the current population, the proposed criterion maintains an elite preserving mechanism over the generations and use them as the basis of the criterion.

8. Other researchers [1, 8] have computed their online stopping criteria at each generation whereas the proposed criterion is estimated for generations to reduce the possibility of convergence to a local optima.

Though, K-it criterion is easy to implement and computationally less expensive, the probability of resulting to a local optimum in K-it criterion is higher than that of the proposed variance based criterion. This is due to the fact that K-it criterion takes into account the information obtained only in K iterations not the information prior to that. In favorable conditions, K-it criterion may stop the algorithm early and reduce the computation, but in general, and in the worst case scenario, the proposed criterion would allow the algorithm to execute more number of iterations and permit it to converge to a global optimum solution.

8. Conclusion and Scope for Future Work

A new stopping criterion for EGA has been suggested here. It has been shown that the variance of the best solutions obtained so far tends to 0 as $n \rightarrow \infty$. In practice, a user needs to suggest an appropriate value for the upper bound of the variance for his/her problem. It is experimentally found that different problems with the same size of search space need different bounds for variance to obtain global optimal solution. For better accuracy, the user needs to choose sufficiently small value for ϵ (bound for variance). No automatic way of choosing the value ϵ is suggested here. The choice of ϵ depends upon the accuracy the user desires. It may also be desirable for the user to know the number of iterations for obtaining ϵ accuracy in variance before performing the experiment. This is an extremely challenging problem and it is the matter for further research.

Acknowledgement

The authors sincerely thank the reviewers for their invaluable comments and suggestions that helped to improve the article to a great extent. The work was done when Prof. Sankar K. Pal held J. C. Bose National Fellowship of the Government of India.

References

[1] Aytug, H., Koehler, G. J.: New stopping criteria for Genetic Algorithms, *European Journal of Operational Research*, **126**, 2000, 662–674.

[2] Bhandari, D., Murthy, C. A., Pal, S. K.: Genetic Algorithms with Elitist Model and its Convergence, *International Journal of Pattern recognition and Artificial Intelligence*, **10**(6), 1996, 731–747.

[3] Davis, T. E., Principe, C. J.: A simulated annealing like convergence theory for the simple genetic algorithm, *Proceedings of 4th int. conf. on genetic algorithms*, Morgan Kaufmann, Los Altos, CA, 1991.

[4] Dejong, K. A.: *An analysis of the behaviour of a class of genetic adaptive systems*, Ph.D. Thesis, Department of Computer and Communication Science, Univ. of Michigan, Ann Arbor, 1975.

[5] Feller, W.: *An Introduction to Probability Theory and its Applications (Vol. I)*, Wiley Eastern Pvt. Ltd., New Delhi, 1972.

[6] Goldberg, D. E.: *Genetic Algorithms: Search,Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[7] Greenhalgh, D., Marshall, S.: Convergence Criteria for Genetic Algorithms, *SIAM Journal on Computing*, **30**(1), 2000, 269–282.

[8] Jain, B. J., Pohlheim, H., Wegener, J.: On Termination Criteria of Evolutionary Algorithms, *GECCO 2001 - Proceedings of the Genetic and Evolutionary Computation Conference.*, Morgan Kauffmann, San Francisco, CA, 2001.

[9] Kallel, L., Naudts, B., Rogers, A., Eds.: *Theoretical aspects of evolutionary computing*, Springer, Heidelberg, Heidelberg, 2001.

[10] Maity, S. P., Kundu, M. K.: Genetic algorithms for optimality of data hiding in digital images, *Soft Computing*, **13**(4), 2009, 361–373.

[11] Michalewicz, Z.: *Genetic Algorithms + Data Structure = Evolution programs*, Springer Verlag, 1992.

[12] Munteanu, C., Rosa, A.: Gray-Scale Image Enhancement as an Automatic Process Driven by Evolution, *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, **34**(2), 2004, 1292–1298.

[13] Murthy, C. A., Bhandari, D., Pal, S. K.: ϵ-Optimal Stopping Time for Genetic Algorithms, *Fundamenta Informaticae*, **35**(1-4), 1998, 91–111.

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

[15] Pal, S. K., Bhandari, D.: Selection of optimal set of weights in a layered network using Genetic algorithms, *Information Sciences*, **80**(3-4), 1994, 213–234.

[16] Pal, S. K., Bhandari, D., Kundu, M. K.: Genetic algorithms for optimal image enhancement, *Pattern Recognition Letters*, **15**, 1994, 261–271.

[17] Pal, S. K., Ghosh, A., Kundu, M. K., Eds.: *Soft Computing for Image Processing*, Physica Verlag, Heidelberg, 2000.

[18] Rudolph, G.: Convergence analysis of canonical genetic algorithm, *IEEE Transactions on Neural networks*, **5**(1), 1994, 96–101.

[19] Safe, M., Carballido, J. A., Ponzoni, I., Brignole, N. B.: On stopping criteria for genetic algorithms, *Proc. of Advances in artificial intelligenceSBIA 2004, 17th Brazilian symposium on artificial intelligence* (A. L. C. Bazzan, S. Labidi, Eds.), Springer, Berlin, 2004.

[20] Suzuki, J.: A Markov chain analysis on a genetic algorithm, *IEEE Transactions on Systems, Man and Cybernetics*, **25**(4), 1995, 655–659.

[21] Vose, M. D.: Modeling simple genetic algorithms, *Foundations of genetic algorithms II* (D. Whitley, Ed.), Morgan Kaufmann Publishers, 1993.