Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 193 (2007) 211–230

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

A new mutation operator for real coded genetic algorithms

Kusum Deep, Manoj Thakur *

Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee 247 667, India

Abstract

In this paper, a new mutation operator called power mutation (PM) is introduced for real coded genetic algorithms (RCGA). The performance of PM is compared with two other existing real coded mutation operators taken from literature namely: non-uniform mutation (NUM) and Makinen, Periaux and Toivanen mutation (MPTM). Using the various combinations of two crossovers (Laplace crossover [Kusum Deep, Manoj Thakur, A new crossover operator for real coded genetic algorithms, Applied Mathematics and Computations, accepted for publication, doi:10.1016/j.amc.2006.10.047] and Heuristic crossover [Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1992; A.H. Wright, Genetic algorithms for real parameter optimization, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms I, Morgan Kaufmann, San Mateo, 1991, pp. 205–218]) and three mutation operators (the newly defined mutation in this paper, PM, NUM and MPTM) six generational real coded GAs are compared on a set of 20 benchmark global optimization test problems. Various performance criterion are used to judge the *efficiency*, *accuracy* and *reliability* of all the RCGAs. The results show that the RCGA using the proposed power mutation, when used in conjunction with the earlier defined Laplace crossover, outperforms all other GAs considered in this study.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Genetic algorithms; Global optimization; Real coded mutation operator

1. Introduction

Many real life problems can be modeled as continuous nonlinear optimization problems and often their global optimal solution is sought. A typical nonlinear global optimization problem is defined as

$$\text{Maximize/Minimize } f(x), \quad \text{where } x \in S \subseteq R^n.$$

S being the search space. The problem is to seek that value of x from within S, say x^* which minimizes/maximizes $f(x)$.

The optimization techniques available in literature for solving general nonlinear optimization problems could be classified into two groups:

* Corresponding author.
E-mail address: manojdma@iitr.ernet.in (M. Thakur).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.03.046

- Deterministic optimization techniques.
- Stochastic optimization techniques.

Deterministic optimization techniques work with a single solution (point by point algorithms) and are local minimizer in nature because they begin the search procedure with a guess solution (often chosen randomly in the search space) and if this guess solution is not chosen close enough to the global minimum solution, they are likely to be trapped in the local minimum solution. Most of them are designed to solve a particular class of optimization problems.

On the other hand, stochastic optimization techniques like evolutionary algorithms, simulated annealing, etc. rely heavily on computational power. Among these, evolutionary algorithms are found to be very promising global optimizers. Evolutionary algorithms consist of three population based heuristic methodologies: genetic algorithms, evolutionary programming, and evolutionary strategies. Genetic algorithms (GA) are perhaps the most popular evolutionary algorithms [1].

Genetic algorithms mimic the principle of survival of the fittest laid by Charles Darwin. In GAs, firstly, an encoding scheme is used to represent a point (individual) in the search of decision variables and then each individual of the population is assigned its fitness based on some criteria. Then, repeatedly a series of generations are carried out which consists of the three major operators, namely selection, crossover and mutation.

In earlier implementations [2,3] the decision variables were encoded as strings of binary alphabets *zero* and *one*. The performance of binary GAs are found to be satisfactory on small and moderate size problem requiring less precision in the solution but for high dimensional problems in which higher degree of precision is desired, binary GAs require huge computational time and memory (Goldberg [4]).

To overcome these difficulties real coded GAs, in which decision variables are encoded as real numbers, are now becoming popular. It has now been established that real coded GAs are superior to binary coded GAs for continuous optimization problems [5]. In real coded GAs crossover has always been considered as a fundamental search operator. Most of the research in this area is concentrated upon constructing new crossover operators to effectively blend the genetic information of the parents and improving the way of applying existing crossover operators in the best possible way. A variety of crossover operators are proposed in the literature to design both steady state GAs [6–10] as well as generational GAs [11–20].

In [21] Deep and Thakur introduce a new parent centric real coded crossover operator, called the Laplace crossover (LX). LX is used in conjunction with two well known mutation operators namely the non-uniform mutation (NUM) [11] and Makinen, Periaux and Toivanen mutation (MPTM) [22] to define two new generational genetic algorithms LX-MPTM and LX-NUM. These two genetic algorithms are compared with two existing generational genetic algorithms (HX-MPTM and HX-NUM) which comprise of Heuristic crossover operator and same two mutation operators. A set of 20 test problems available in the global optimization literature is used to test the performance of these four genetic algorithms. To judge the performance of the LX operator, two kinds of analysis is performed. Firstly a pairwise comparison is performed between LX-MPTM and HX-MPTM, and then between LX-NUM and HX-NUM. Secondly the overall comparison of performance of all the four genetic algorithms is carried out based on a performance index (PI). The comparative study shows that Laplace crossover (LX) performs quite well and one of the genetic algorithms defined (LX-MPTM) outperforms other remaining genetic algorithms considered.

In GA literature relatively less effort has been put into designing new mutation operators for real coded GAs. In an endeavor to define new operators for real coded genetic algorithms, in this paper, a new mutation operator called the power mutation (PM) is proposed. Based on PM, two new generational real coded GAs called LX–PM and HX–PM are constructed, which make use of Laplace crossover with power mutation and Heuristic crossover with power mutation, respectively. In order to establish the strength of power mutation an extensive computational analysis is performed and discussed.

The paper is organized as follows. In Section 2, the literature review on real coded mutation operators is given. Section 3 defines the proposed mutation operator called power mutation (PM), while in Section 4 the other operators used in the present study are stated. Two new GAs using PM are described in Section 5. The list of test problems is provided in Section 6. Sections 7 comprise of the experimental setup, numerical results, statistical tests and discussion of results is explained. In Section 8, the conclusions from the present study are drawn.

2. Literature review on mutation operators

Mutation operator provides a random diversity in the population as described in [23]. Apart from designing new mutation operators, researchers have investigated numerous ways of applying mutation operators. While applying a mutation operator two issues are of paramount importance: firstly, the proportion of population undergoing mutation, i.e. the probability of applying mutation operator and secondly, the strength of mutation, i.e. the perturbation produced in a chromosome. A variety of such schemes can be found in Herrera [24,25]. In another approach Krasnogor and Smith [26] provide a discrete set of mutation rates to the RCGA and its advantages are discussed in [27].

Michalewicz [11] proposed random (uniform) mutation and non-uniform mutation. In random mutation, where a gene is replaced with a random value between its lower and upper bounds. On the other hand, in non-uniform mutation the step size decreases as the generations increase, thus making a uniform search in the initial space and very little at the later stage. A special case of uniform mutation is boundary mutation [28] in which a gene is replaced by either its lower bound or upper bound.

Muhlenbein's BGA mutation [29] creates a solution in the neighborhood of a chromosome. The probability of producing a solution on either sides of a chromosome is equal and the strength of mutation is usually set to be 10% of the allowed range of the particular gene. Based on a similar idea Voigt proposed discrete modal mutation and continuous modal mutation [30].

In Gaussian mutation operator two parameters: the mean (usually set to zero), and the standard deviation of the Gaussian distribution are required. The standard deviation of the Gaussian distribution dictates the strength of mutation. A Gaussian mutation with self adaptive mutation step size [31] and a self adaptive Gaussian mutation with feed back based adaptation of population size is proposed in [32].

A co-evolving set of simple directions for each chromosome govern the result of gene mutations in pointed directed (PoD) mutation [33]. Another directed mutation based on the momentum notion (used to accelerate the gradient descent training of neural networks) is proposed in [34]. Both PoD and hybrid system combining the momentum-based and standard mutation operators are compared against Gaussian mutation on a small set of unconstrained benchmark problems and reported to be advantageous over Gaussian mutation.

Munteanu and Lazarescu [35] proposed PCA-mutation. This mutation operator is motivated by principal component analysis (PCA) used in statistics. PCA-mutation is observed to produce higher level of diversity in the population as compared to uniform mutation and non-uniform mutation on IIR Filter design problem.

Polynomial mutation operator [19,36] is one of the most widely used mutation operator and has been applied successfully in solving single and multi-objective optimization problems [36].

Makinen, Periaux and Toivanen mutation [22] is a relatively new mutation operator and has been applied to solve multidisciplinary shape optimization problem as well as a large set of constrained optimization problems [40].

Ling and Leung [20] proposed wavelet mutation which is based on wavelet theory. Wavelet mutation uses Morlet wavelet as the mother wavelet. The performance of average bound crossover and wavelet mutation based GA is reported better than some other real coded GAs on a suite of benchmark test functions as well as economic load dispatch and tuning an associative-memory neural network.

3. The proposed power mutation operator

The proposed mutation operator is based on power distribution. We name it as the power mutation (PM). Its distribution function is given by

$$f(x) = px^{p-1}, \quad 0 \leqslant x \leqslant 1 \tag{1}$$

and the density function is given by

$$F(x) = x^p, \quad 0 \leqslant x \leqslant 1, \tag{2}$$

where p is the index of the distribution. The PM is used to create a solution y in the vicinity of a parent solution $\bar{x}$ in the following manner. First a uniform random number t between 0 and 1 is created and a random

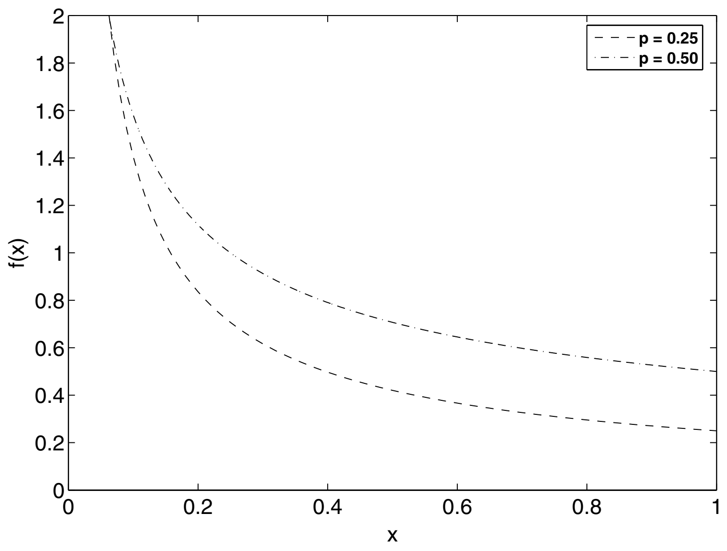

Fig. 1. Density function of power distribution (for $p = 0.25$ and $p = 0.50$).

number s is created which follows the above mentioned distribution. Then following formula is used to create the muted solution

$$y = \begin{cases} \bar{x} - s(\bar{x} - x^{\mathrm{l}}) & \text{if } t < r, \\ \bar{x} + s(x^{\mathrm{u}} - \bar{x}) & \text{if } t \geqslant r, \end{cases} \tag{3}$$

where $t = \frac{\bar{x} - x^{\mathrm{l}}}{x^{\mathrm{u}} - x^{\mathrm{l}}}$ and x^{l} and x^{u} are lower and upper bounds of the decision variable and r is a uniformly distributed random number between 0 and 1. The distribution function of the power distribution (for $p = 0.25$ and $p = 0.50$) is shown in Fig. 1. The strength of mutation is governed by the index of the mutation (p). For small values of p less perturbance in the solution is expected and for large values of p more diversity is achieved. The probability of producing a mutated solution y on left (right) side of $\bar{x}$ is proportional to distance of $\bar{x}$ from $x^{\mathrm{l}}(x^{\mathrm{u}})$ and the muted solution is always feasible.

4. The remaining operators used in this study

In this section, we define the crossover and mutation operators used in this paper, i.e. Laplace crossover (LX), Heuristic crossover (HX), Makinen, Periaux and Toivanen mutation (MPTM) and non-uniform mutation (NUM).

4.1. Laplace crossover

Laplace crossover (LX) is recently introduced by Deep and Thakur [21]. It is a parent centric operator. Using LX, two offsprings $y^{(1)} = (y_1^{(1)}, y_2^{(1)}, \ldots, y_n^{(1)})$ and $y^{(2)} = (y_1^{(2)}, y_2^{(2)}, \ldots, y_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)})$ in the following way.

First, two uniformly distributed random numbers $u_i, u_i' \in [0,\ 1]$ are generated. Then, a random number β_i is generated which follows the Laplace distribution by simply inverting the distribution function of Laplace distribution as follows:

$$\beta_i = \begin{cases} a - b\log_e(u_i), & u_i' \leqslant \frac{1}{2}, \\ a + b\log_e(u_i), & u_i' > \frac{1}{2}. \end{cases} \tag{4}$$

The offsprings are given by the equation

$$y_i^{(1)} = x_i^{(1)} + \beta_i |x_i^{(1)} - x_i^{(2)}|, \tag{5}$$

$$y_i^{(2)} = x_i^{(2)} + \beta_i |x_i^{(1)} - x_i^{(2)}|. \tag{6}$$

From the above two equations it is clear that

(i) Both the offsprings are placed symmetrically with respect to the position of the parents.
(ii) For smaller values of b, offsprings are likely to be produced near the parents and for larger values of b offsprings are expected to be produced far from the parents.
(iii) For a fixed value of a and b, LX dispenses offsprings proportional to the spread of parents, i.e. if the parents are near to each other the offsprings are expected to be near to each other and if the parents are far from each other then the offsprings are likely to be far from each other.

This way the LX operator exhibits self adaptive behavior.

If LX produces an offspring which violates a box-constraint, i.e. $x_i < x_i^{\mathrm{l}}$ or $x_i > x_i^{\mathrm{u}}$ for some i, then x_i is assigned a random value in the interval $[x_i^{\mathrm{l}}, x_i^{\mathrm{u}}]$.

4.2. Heuristic crossover

Heuristic crossover (HX) was introduced by Wright [16]. Later, Michalewicz et al. [38] used HX to solve some linearly constrained problems and incorporated HX in GENOCOP system. Subsequently, Michalewicz [39] also applied HX to solve nonlinear constrained optimization problems. Some more experiments with HX were reported in Michalewicz and Schoenauer [40]. In recent studies [37,41], HX is applied to solve constrained as well as unconstrained optimization problems having various levels of difficulty.

From a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)})$ an offspring $y = (y_1, y_2, \ldots, y_n)$ is generated in the following manner:

$$y_i = u(x_i^{(2)} - x_i^{(1)}) + x_i^{(2)}, \tag{7}$$

where u is a uniformly distributed random number in the interval $[0, 1]$ and the parent $x^{(2)}$ has fitness value not worse than that of parent $x^{(1)}$. If the offspring so generated lie outside the feasible region, a new random number u is generated to produce another offspring using Eq. (7). If required, this process is repeated up to k times. If HX fails to produce a feasible offspring after k attempts, a random point in feasible region is chosen in place of the infeasible offspring produced.

There are some interesting features which make HX different from other crossover operators available in real coded GAs literature.

(i) HX produces at most one offspring from a given pair of parents.
(ii) Unlike other real coded crossover operators, HX makes use of fitness function values of parents in producing offspring.

4.3. Makinen, Periaux and Toivanen mutation (MPTM)

This mutation operator was proposed by Makinen et al. [22] in which they used it in a GA to solve some multidisciplinary shape optimization problems in aerodynamics and electromagnetics. In [40] it was used in a GA to solve a large set to solve constrained optimization problem. Although this operator has not been given any specific name in literature, we have named it MPTM on the originators of this operator (Makinen, Periaux and Toivanen).

From a point $x = (x_1, x_2, \ldots, x_n)$ the mutated point $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ is created as follows. Let r_i be a uniformly distributed random number such that $r_i \in [0, 1]$. Then the muted solution is given by

$$\hat{x}_i = (1-\hat{t}_i)x_i^{\mathrm{l}} + \hat{t}_i x_i^{\mathrm{u}}, \tag{8}$$

$$\text{where } \hat{t}_i = \begin{cases} t_i - t_i\left(\frac{t_i - r_i}{t_i}\right)^b & \text{if } r_i < t_i, \\ t_i & \text{if } r_i = t_i, \\ .t_i + (1-t_i)\left(\frac{r_i - t_i}{1-t_i}\right)^b & \text{if } r_i > t_i, \end{cases} \tag{9}$$

$$\text{and} \quad t_i = \frac{x_i - x_i^{\mathrm{l}}}{x_i^{\mathrm{u}} - x_i}, \tag{10}$$

where x_i^{l} and x_i^{u} are lower and upper bounds of the *i*th decision variable, respectively.

Unlike non-uniform mutation operator, the strength of MPTM does not decrease as the generation increases.

4.4. Non-uniform mutation (NUM)

Michalewicz's non-uniform mutation is one of the most widely used mutation operators in real coded GAs [11,37–39]. From a point $x^t = (x_1^t, x_2^t, \ldots, x_n^t)$ the muted point $x^{t+1} = (x_1^{t+1}, x_2^{t+1}, \ldots, x_n^{t+1})$ is created as follows:

$$x_i^{t+1} = \begin{cases} x_i^t + \Delta(t, x_i^{\mathrm{u}} - x_i^t), & \text{if } r \leqslant 0.5, \\ x_i^t - \Delta(t, x_i^t - x_i^{\mathrm{l}}), & \text{otherwise}, \end{cases} \tag{11}$$

where *t* is current generation number and *r* is a uniformly distributed random number between 0 and 1. x_i^{l} and x_i^{u} are lower and upper bounds of the *i*th component of the decision vector, respectively. The function $\Delta(t, y)$ given below takes value in the interval $[0, y]$.

$$\Delta(t, y) = y\left(1 - u^{\left(1-\frac{t}{T}\right)^b}\right), \tag{12}$$

where *u* is a uniformly distributed random number in the interval $[0, 1]$, *T* is the maximum number of generations and *b* is a parameter, determining the strength of the mutation operator. In the initial generations non-uniform mutation tends to search the space uniformly and in the later generations it tends to search the space locally, i.e. closer to its descendants [11].

5. The proposed new RCGAs

The objective of this study is to introduce a new mutation operator PM and to evaluate its performance in comparison to other mutation operators existing in literature namely MPTM and NUM. Therefore, two new RCGAs are designed namely, LX-PM and HX-PM which combine the use of Laplace crossover and Heuristic crossover with power mutation, respectively. They are compared with the obvious choice of GAs namely the four existing GAs LX-MPTM, LX-NUM and HX-MPTM, HX-NUM. which make use of Laplace crossover (LX), Heuristic crossover (HX), Makinen, Periaux and Toivanen mutation (MPTM) and non-uniform mutation (NUM). Table 1 summarizes the operators used in all the six GAs used in this study.

Table 1
Summary of operators used in the six GAs

Group	Name of GA	Crossover	Mutation	Selection
Group 1	LX-PM	LX	PM	Tournament
	LX-MPTM	LX	MPTM	Tournament
	LX-NUM	LX	NUM	Tournament
Group 2	HX-PM	HX	PM	Tournament
	HX-MPTM	HX	MPTM	Tournament
	HX-NUM	HX	NUM	Tournament

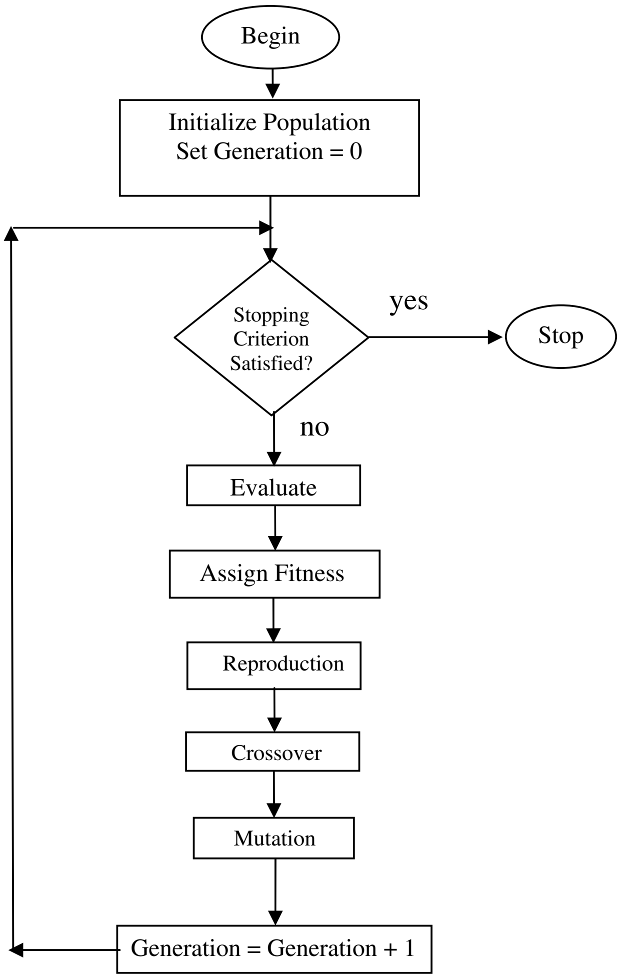

Fig. 2. Flow chart of all GAs used for comparative study.

The structure of all the genetic algorithms defined above is very much similar to the Simple GA of Goldberg (Fig. 2).

6. The test bed

The test bed chosen consists of a set of 20 benchmark test problems of different levels of complexity and multimodality from the global optimization literature. This set contains unimodal as well multimodal problems. All the problems are scalable, i.e. the number of decision variables can be increased/decreased as per users choice. The corrected formulation of the test problems is given below

1. *Ackley's Problem*

$$\min_x f(x) = -20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \quad -30 \leqslant x_i \leqslant 30,$$

$x^* = (0, 0, \ldots, 0)$ and $f(x^*) = 0$.

2. *Cosine Mixture Problem*

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n} \cos(5\pi x_i) \quad -1 \leqslant x_i \leqslant 1,\ x^* = (0,0,\ldots,0) \text{ and } f(x^*) = -0.1n.$$

3. *Exponential Problem*

$$\min_x f(x) = -\left(\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)\right) \quad -1 \leqslant x_i \leqslant 1,\ x^* = (0,0,\ldots,0) \text{ and } f(x^*) = -1.$$

4. *Griewank Problem*

$$\min_x f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad -600 \leqslant x_i \leqslant 600,\ x^* = (0,0,\ldots,0) \text{ and } f(x^*) = 0.$$

5. *Levy and Montalvo Problem 1*

$$\min_x f(x) = \frac{\pi}{n}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right),$$

where $y_i = 1 + \frac{1}{4}(x_i + 1), \quad -10 \leqslant x_i \leqslant 10,\ x^* = (-1,-1,\ldots,-1)$ and $f(x^*) = 0$.

6. *Levy and Montalvo Problem 2*

$$\min_x f(x) = 0.1\left(\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right)\right)$$

$$-5 \leqslant x_i \leqslant 5,\ x^* = (1,1,\ldots,1) \text{ and } f(x^*) = 0.$$

7. *Paviani Problem*

$$\min_x f(x) = \sum_{i=1}^{n}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - \left(\prod_{i=1}^{10} x_i\right)^{0.2} \quad 2 \leqslant x_i \leqslant 10,\ f(x^*) \approx -997807.705158.$$

8. *Rastrigin Problem*

$$\min_x f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)] \quad -5.12 \leqslant x_i \leqslant 5.12,\ x^* = (0,0,\ldots,0) \text{ and } f(x^*) = 0.$$

9. *Rosenbrock Problem*

$$\min_x f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad -30 \leqslant x_i \leqslant 30,\ x^* = (1,1,1,\ldots.,1) \text{ and } f(x^*) = 0.$$

10. *Schwefel Problem*

$$\min_x f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \quad -500 \leqslant x_i \leqslant 500,\ x^* = (420.9867, 420.9867,\ldots,420.9867)$$

and $f(x^*) = 0$.

11. *Sinusoidal Problem*

$$\min_x f(x) = -\left[2.5\prod_{i=1}^{n} \sin\left(x_i - \frac{\pi}{6}\right) + \prod_{i=1}^{n} \sin\left(5\left(x_i - \frac{\pi}{6}\right)\right)\right] \quad 0 \leqslant x_i \leqslant \pi,$$

$x^* = \left(\frac{2\pi}{3}, \frac{2\pi}{3}, \ldots, \frac{2\pi}{3}\right)$ and $f(x^*) = -3.5$.

12. *Zakharov's Function*

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} \frac{i}{2} x_i\right)^2 + \left(\sum_{i=1}^{n} \frac{i}{2} x_i\right)^4 \quad -5.12 \leqslant x_i \leqslant 5.12,$$

$x^* = (0, 0, \ldots, 0)$ and $f(x^*) = 0$.

13. *Sphere Function*

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 \quad -5.12 \leqslant x_i \leqslant 5.12,\ x^* = (0, 0, \ldots, 0) \text{ and } f(x^*) = 0.$$

14. *Axis Parallel Hyper Ellipsoid*

$$\min_x f(x) = \sum_{i=1}^{n} \mathrm{i}x_i^2 \quad -5.12 \leqslant x_i \leqslant 5.12,\ x^* = (0, 0, \ldots, 0) \text{ and } f(x^*) = 0.$$

15. *Schewefel Problem 3*

$$\min_x f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \quad -10 \leqslant x_i \leqslant 10,\ x^* = (0, 0, \ldots, 0) \text{ and } f(x^*) = 0.$$

16. *Schewefel Problem 4*

$$\min_x f(x) = \max_i \{|x_i|, 1 \leqslant i \leqslant n\} \quad -100 \leqslant x_i \leqslant 100,\ x^* = (0, 0, \ldots, 0) \text{ and } f(x^*) = 0.$$

17. *De-Jong's Function with noise*

$$\min_x f(x) = \sum_{i=1}^{n} (x_i^4 + r \text{ and } (0, 1)) \quad -10 \leqslant x_i \leqslant 10,\ x^* = (0, 0, \ldots, 0) \text{ and } f(x^*) = 0.$$

18. *Ellipsoidal Function*

$$\min_x f(x) = \sum_{i=1}^{n} (x_i - i)^2 \quad -n \leqslant x_i \leqslant n,\ x^* = (1, 2, \ldots, n) \text{ and } f(x^*) = 0.$$

19. *Generalized Penalized Function 1*

$$\min_x f(x) = \frac{\pi}{n}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right) + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

where $y_i = 1 + \frac{1}{4}(x_i + 1) \quad -50 \leqslant x_i \leqslant 50,\ x^* = (-1, -1, \ldots, -1)$ and $f(x^*) = 0$.

20. *Generalized Penalized Function 2*

$$\min_x f(x) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\right)$$

$$+ \sum_{i=1}^{n} u(x_i, 10, 100, 4) \quad -50 \leqslant x_i \leqslant 50,\ x^* = (1, 1, \ldots, 1) \text{ and } f(x^*) = 0.$$

In problem number 19 and 20, the value of penalty function u is given by the following expression

$$u(x,a,k,m) = \begin{cases} k * \mathrm{pow}((x-a),m) & \text{if } x > a, \\ -k * \mathrm{pow}((x-a),m) & \text{if } x < -a, \\ 0 & \text{otherwise.} \end{cases}$$

7. Results and discussions

For both the newly defined RCGAs, namely LX-PM and HX-PM, an extensive experimental study of various possible combinations of crossover probability, mutation probability and the indices occurring in the expressions of crossover and mutation is done. This fine tuning of parameters is carried out for each of the two GAs, so that for a fair comparison, each GA with the optimal parameter setting is used for comparison. The optimal parameters settings for the remaining four GAs namely HX-MPTM, LX-MPTM, HX-NUM and LX-NUM are picked up from [21]. The final parameter settings for all the RCGAs is given in Table 2. We do not claim that these parameter settings are the best for any problem in general, but these values are selected and recommended since they are found to be repeatedly giving good results for most of the problems and hence they are appropriate values to choose if we talk about the overall performance of the algorithms in general.

For a uniform testing environment of all the GAs, the population size is taken to be ten times the number of decision variables. All the algorithms use tournament selection. In addition all the GAs are elite preserving GAs with elitism size one, i.e. if the best individual of the kth population is better that the best individual in $(k+1)$th population, then it replaces the best individual of the $(k+1)$th population. The value of a for LX is fixed to be zero and for HX, k is fixed to be 4 (as recommended in [37]).

All the algorithms are implemented in C++ and the experiments are done on a PIV 2.8 GHz machine with 512 MB RAM under WINXP platform. The number of variables in the entire problem set is fixed to be 30. Each GA has been run 30 times with different initial populations.

In GA literature the performance of a GA is usually measured on the basis of three criteria viz. *reliability, efficiency* and *accuracy* of the algorithm. The *reliability* assess that how frequently in a fixed number of independent runs a GA converges the near optimal solution, the*efficiency* of a GA is the measure of the rate of convergence, and *accuracy* signifies the degree of precision in locating global minima. For this, we have recorded the percentage of success; execution time and average number of function evaluations of successful runs as well as the mean and standard deviation of objective function values. For a particular problem and a particular GA, a run is said to be a successful run if the best objective function value found in that run lies within 1% accuracy of the best know objective function value of that problem. The maximum numbers of generations are fixed to be 5000 for all the GAs.

The six GAs are divided into two groups (three GAs in each group). The first group consists of LX-PM, LX-MPTM and LX-NUM whereas the second group consists of HX-PM, HX-MPTM and HX-NUM. The motto behind this kind of division is that we are interested in analyzing the behavior of the newly defined mutation operator, namely PM. It is evident from the above division that in both the groups the crossover

Table 2
Parameter values for the six GAs

Group	Name of GA	Crossover probability	Mutation probability	Crossover index	Mutation index	Tournament size
Group 1	LX-PM	0.55	0.005	0.35	0.25	3
	LX-MPTM	0.50	0.005	0.20	4	2
	LX-NUM	0.50	0.005	0.15	4	2
Group 2	HX-PM	0.55	0.010	–	0.20	3
	HX-MPTM	0.70	0.020	–	4	3
	HX-NUM	0.70	0.010	–	4	3

operator is same (LX in Group 1 and HX in Group 2) but mutation operators are different. So a relative comparison of GAs from the same group will help us to judge the performance of new mutation operator PM as against MPTM and NUM. The comparison is done to measure the performance in two different ways as explained below.

7.1. *First method of analysis*

7.1.1. *LX-PM vs LX-MPTM and LX-NUM*

The average number of function evaluations of successful runs, the average execution time of successful runs and the number of successful runs (out of 30 runs), for GAs in Group 1 (LX-PM, LX-MPTM and LX-NUM) are listed in Table 3. From this table it is observed that both LX-PM and LX-MPTM are having identical success rate on all the problems. For problem nos. 2, 10 and 19 LX-NUM gives slightly less success (27, 28, 29 out of 30 runs, respectively) than LX-PM and LX-MPTM and it is completely unable to solve problem no. 8. Further, none of the GAs of Group 1 is able to solve problem no. 9 (Rosenbrock's function). In 17 out of 19 problems, where LX-PM and LX-MPTM have identical success rate, LX-PM require less function evaluation and less execution time to converge near global minima. LX-PM and LX-NUM have same success rate in 15 problems and in 14 instances LX-PM is better than LX-NUM both in terms of average function evaluation and less execution time of successful runs. There are three cases (problem nos. 2, 10 and 19) where LX-PM completely outperforms LX-NUM on all the three criteria.

Like in Deep and Thakur [21], in order to compare the quality of the solutions found, we list the mean and standard deviation of the best objective function values found after a fixed number of generations for each test case in Table 4. In sixteen problems the mean objective function values and corresponding standard deviations found by LX-PM is found to be less than LX-MPTM and in 14 cases average objective function values and corresponding standard deviations found by LX-PM is found to be less than that of LX-NUM.

We apply *t*-test at a 0.05 level of significance as proposed in [24] and elsewhere in the literature to ascertain the significance of the difference of the mean of the objective function values. A '∼' sign indicates that there is no significant difference in the means and a '+' ('−') sign indicates that the mean objective function value

Table 3
No. of successful runs, average no. of function evaluations and execution time of successful runs for three GAs of Group 1

Problem number	Number of successful runs (out of 30 runs)			Average number of function evaluations of successful runs			Average time of execution of successful runs (s)		
	LX-PM	LX-MPTM	LX-NUM	LX-PM	LX-MPTM	LX-NUM	LX-PM	LX-MPTM	LX-NUM
1	30	30	30	86,661	116,731	130,041	1.625	2.243	2.473
2	30	30	27	34,471	46,601	54,723	0.611	0.847	0.997
3	30	30	30	23,191	31,221	35,141	0.277	0.389	0.437
4	30	30	30	100,001	143,371	159,971	1.851	2.723	3.035
5	30	30	30	28,251	36,361	46,761	0.736	0.963	1.245
6	30	30	30	40,241	51,091	68,721	0.985	1.273	1.727
7	30	30	30	83,651	153,161	122,071	1.608	3.017	2.409
8	30	30	0	165,471	350,541	–	2.997	6.515	–
9	0	0	0	–	–	–	–	–	–
10	30	30	29	99,141	236,411	245,918	1.658	4.081	4.240
11	30	30	30	41,721	57,381	61,951	1.100	1.541	1.666
12	30	30	30	63,271	65,711	78,681	1.563	1.642	1.969
13	30	30	30	43,541	59,861	67,661	0.505	0.724	0.819
14	30	30	30	56,281	78,571	89,391	0.651	0.944	1.075
15	30	30	30	71,331	105,721	118,541	0.997	1.530	1.714
16	30	30	30	730	720	690	0.007	0.008	0.007
17	30	30	30	861	771	911	0.017	0.015	0.018
18	30	30	30	71,001	101,331	104,281	0.821	1.220	1.254
19	30	30	29	72,651	90,121	111,322	2.066	2.595	3.206
20	30	30	30	93,321	119,521	140,751	2.542	3.293	3.887

Table 4
Mean and standard deviation of objective function values and *t*-test results for mean objective function values for three GAs of Group 1

Problem Number	Mean			Standard deviation			*t*-Test results	
	LX-PM	LX-MPTM	LX-NUM	LX-PM	LX-MPTM	LX-NUM	LX-PM vs. LX-MPTM	LX-PM vs. LX-NUM
1	1.01E−10	2.18E−07	6.66E−07	1.04E−10	6.94E−08	2.80E−07	+	+
2	−3.00E+00	−3.00E+00	−2.99E+00	0.00E+00	0.00E+00	4.51E−02	~	+
3	−1.00E+00	−1.00E+00	−1.00E+00	1.73E−08	1.74E−06	7.13E−06	+	+
4	1.94E−03	1.14E−03	8.57E−04	1.57E−03	1.18E−03	5.36E−04	−	−
5	3.20E−19	9.94E−14	3.34E−09	2.89E−19	6.45E−14	1.83E−08	+	~
6	7.95E−23	8.56E−16	3.91E−09	9.92E−23	1.29E−15	2.14E−08	+	~
7	−9.98E+05	−9.93E+05	−9.97E+05	5.59E+00	1.08E+03	5.33E+02	+	+
8	0.00E+00	1.23E−12	7.30E+00	0.00E+00	4.89E−12	3.08E+00	~	+
9	1.58E+01	1.85E+01	1.85E+01	2.15E+00	6.38E−01	1.80E−01	+	+
10	−1.26E+04	−1.26E+04	−1.25E+04	1.85E−12	9.32E−05	9.41E+01	~	+
11	−3.50E+00	−3.50E+00	−3.50E+00	2.98E−08	4.29E−06	2.54E−05	+	+
12	1.95E−20	8.04E−15	3.32E−10	6.83E−20	1.20E−14	1.82E−09	+	~
13	4.75E−11	1.32E−07	8.92E−07	3.34E−11	7.44E−08	5.48E−07	+	+
14	2.82E−12	3.94E−08	3.43E−07	2.09E−12	2.31E−08	1.34E−07	+	+
15	3.03E−08	4.28E−05	1.32E−04	1.31E−08	1.40E−05	4.09E−05	+	+
16	3.60E−05	5.35E−06	1.23E−05	1.70E−04	1.55E−05	3.51E−05	~	~
17	2.32E−04	1.63E−04	1.38E−04	1.95E−04	1.30E−04	1.20E−04	~	−
18	9.26E−11	1.11E−06	1.98E−06	1.03E−10	7.05E−07	1.02E−06	+	+
19	9.48E−32	1.75E−22	3.46E−03	1.70E−31	0.00E+00	1.89E−02	+	~
20	6.07E−31	2.55E−27	5.75E−26	2.24E−30	0.00E+00	0.00E+00	+	+

obtained by LX-PM (LX-MPTM or LX-UM) is significantly large than the mean objective function value obtained LX-MPTM or LX-NUM (LX-PM). Please see (last two column of Table 4).

In five cases there is no significant difference in the mean objective function values of LX-PM and LX-MPTM, in 14 cases the mean objective function value by LX-PM is significantly less than LX-MPTM and there is only one case where the mean objective function value by LX-PM is significantly greater than LX-MPTM.

In 13 cases the mean function values of LX-PM is significantly less than that of LX-NUM, in five cases there is no significant difference in the mean objective function values of LX-PM and LX-NUM. There are only two instances where the mean function value found by LX-PM is significantly greater than LX-NUM.

From the above discussion based on Tables 3 and 4, it is clear that LX-PM outperforms LX-MPTM as well as LX-NUM both in terms of average number of function evaluations of successful runs as well as the average execution time of successful runs. For most of the problems, the mean of objective function value obtained by LX-PM is better than that of LX-MPTM and LX-NUM with less standard deviation. This indicates that LX-PM repeatedly converges to near global optimum in lesser number of generation and time.

7.1.2. HX-PM vs HX-MPTM and HX-NUM

The number of successful runs, the average execution time and the average number of function evaluations of successful runs for GAs in Group 2, (HX-PM, HX-MPTM and HX-NUM) are listed in Table 5. From Table 5 it is evident that except problem number 9 (where HX-MPTM has 0 success rate), HX-PM and HX-MPTM solve all other problems in all the 30 runs.

Tough Rosenbrock's function is coming out to be difficult to solve (as we observed in Group 1 also), yet HX-PM and HX-NUM are partially successful in solving it. This is the only problem where HX-NUM has better success than HX-PM and on the rest of 19 problems HX-PM is either having the same success or better than HX-NUM. It is also observed from Table 5 that in twelve out of 19 problems where both HX-PM and HX-MPTM have similar success, the average function evaluations by HX-PM is less than HX-MPTM and in comparison of HX-NUM; HX-PM takes less function evaluations and execution time in 17 out of 20 cases.

Table 6 comprises the mean and standard deviation of the best objective function values found after a fixed number of generations for each test case. In 14 problems the mean objective function values by HX-PM is

Table 5
Number of successful runs, average no. of function evaluations and execution time of successful runs for three GAs of Group 2

Problem number	Number of successful runs (out of 30 runs)			Average number of function evaluations of successful runs			Average time of execution of successful runs (s)		
	HX-PM	HX-MPTM	HX-NUM	HX-PM	HX-MPTM	HX-NUM	HX-PM	HX-MPTM	HX-NUM
1	30	30	30	178,301	196,401	232,901	2.800	3.565	4.134
2	30	30	27	64,321	63,691	304,671	0.916	1.077	5.207
3	30	30	30	48,091	44,271	60,651	0.411	0.497	0.681
4	30	30	30	236,391	253,931	252,571	3.554	4.486	4.457
5	30	30	24	52,911	51,741	354,201	1.121	1.231	8.570
6	30	30	30	70,661	68,001	92,131	1.476	1.601	2.172
7	30	30	30	70,471	75,751	31,461	1.123	1.452	0.608
8	30	30	28	167,851	245,281	426,651	2.441	4.219	7.358
9	2	0	12	669,751	–	1,368,691	5.453	–	14.621
10	30	30	30	75,181	86,421	66,201	0.956	1.340	1.024
11	30	30	29	81,889	79,071	142,751	1.956	2.092	3.774
12	30	30	28	92,891	94,701	210,791	1.959	2.243	5.060
13	30	30	30	89,751	87,031	107,581	0.733	0.944	1.165
14	30	30	30	117,111	117,911	140,451	0.953	1.278	1.521
15	30	30	30	141,581	158,411	148,251	1.496	2.099	1.966
16	30	30	30	759	679	719	0.007	0.007	0.008
17	30	30	30	811	811	811	0.015	0.017	0.017
18	30	30	24	138,281	143,231	415,081	1.115	1.531	4.412
19	30	30	30	114,281	117,031	230,711	2.703	3.073	6.135
20	30	30	30	150,381	157,521	267,741	3.574	4.151	7.129

Table 6
Mean and standard deviation of objective function values and *t*-test results for mean objective function values for three GAs of Group 2

Problem number	Mean			Standard deviation			*t*-Test results	
	HX-PM	HX-MPTM	HX-NUM	HX-PM	HX-MPTM	HX-NUM	HX-PM vs. HX-MPTM	HX-PM vs. HX-NUM
1	1.01E−10	3.17E−04	1.28E−03	4.59E−05	1.46E−04	1.39E−03	+	+
2	−3.00E+00	−2.99E+00	−2.98E+00	2.85E−14	4.47E−13	5.11E−02	+	+
3	−1.00E+00	−0.99E+00	−9.99E−01	8.40E−05	5.45E−05	3.86E−04	−	+
4	1.94E−03	1.52E−03	3.68E−03	1.38E−03	1.18E−03	3.73E−03	~	+
5	3.20E−19	5.34E−09	3.11E−02	8.91E−10	3.42E−09	8.67E−02	+	+
6	7.95E−23	6.58E−10	1.43E−08	1.42E−10	5.24E−10	7.38E−08	+	~
7	−9.98E+05	−9.98E+05	−9.97E+05	4.63E+00	4.81E+01	1.86E+02	+	+
8	0.00E+00	9.22E−12	3.11E−13	1.03E−13	1.53E−11	5.01E−13	+	+
9	1.84E+01	1.61E+01	1.25E+01	1.93E+01	1.81E+01	4.38E+00	~	−
10	−1.26E+04	−1.26E+04	−1.26E+04	2.01E−08	2.77E−07	7.49E−09	+	−
11	−3.50E+00	−3.50E+00	−3.41E+00	9.47E−01	6.31E−04	4.56E−01	−	~
12	1.95E−20	7.86E−09	2.86E−01	9.40E−10	1.29E−08	1.55E+00	+	~
13	4.75E−11	5.48E−05	3.64E−04	3.70E−05	3.32E−05	2.69E−04	~	+
14	2.82E−12	8.83E−05	4.67E−04	7.01E−05	4.32E−05	3.28E−04	~	+
15	3.03E−08	2.88E−03	1.73E−03	4.90E−04	1.26E−03	1.24E−03	+	~
16	3.60E−05	1.19E−22	2.88E−04	1.04E−15	0.00E+00	1.89E−04	~	+
17	2.32E−04	3.17E−04	8.20E−05	2.17E−04	3.31E−04	2.91E−04	~	−
18	9.26E−11	5.02E−04	3.20E+00	2.17E−04	2.93E−04	6.25E+00	~	+
19	9.48E−32	1.30E−12	7.09E−11	4.72E−14	1.29E−12	1.57E−10	+	+
20	6.07E−31	1.30E−13	5.66E−14	1.59E−15	2.12E−13	9.31E−14	+	+

found to be less than HX-MPTM and in 11 out of these 14 problems standard deviations is also less, in 14 cases average objective function values. On the other hand, in 13 problems the mean objective function values as well as standard deviations of HX-PM is found to be less than HX-NUM.

The *t*-test results at a 0.05 level of significance are shown in last two columns of Table 6. In seven cases there is no significant difference in the mean objective function values of HX-PM and HX-MPTM, in 11 cases the mean objective function value by HX-PM is significantly less than HX-MPTM and there are only two cases where the mean objective function value by HX-PM is significantly greater than HX-MPTM.

In 13 cases the mean objective function values of HX-PM are significantly less than that of HX-NUM, in four cases there is no significant difference in the mean objective function values of HX-PM and HX-NUM. There are three instances where the mean objective function value found by HX-PM is significantly greater than HX-NUM.

This indicates that HX-PM outperforms both HX-MPTM as well as HX-NUM in all the criterion, i.e. the average number of function evaluations of successful runs as well as the average execution time required. Also, the mean of objective function value obtained by HX-PM is better than that of HX-MPTM and HX-NUM with less standard deviation, indicating that HX-PM repeatedly converges to the global optimum in lesser number of generation and time.

7.2. Second method of analysis

To compare the relative performance of all the six GAs simultaneously, we use the same performance index (PI) as used in Deep and Thakur [22,42,43]. The relative performance of an algorithm using this PI is calculated in the following manner.

$$\mathrm{PI} = \frac{1}{N_\mathrm{p}} \sum_{i=1}^{N_\mathrm{p}} (k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i),$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$,

$$\alpha_2^i = \begin{cases} \frac{\mathrm{Mt}^i}{\mathrm{At}^i}, & \text{if } \mathrm{Sr}^i > 0 \\ 0, & \text{if } \mathrm{Sr}^i = 0 \end{cases} \quad \text{and}$$

$$\alpha_3^i = \begin{cases} \frac{\mathrm{Mf}^i}{\mathrm{Af}^i}, & \text{if } Sr^i > 0 \\ 0, & \text{if } Sr^i = 0 \end{cases}.$$

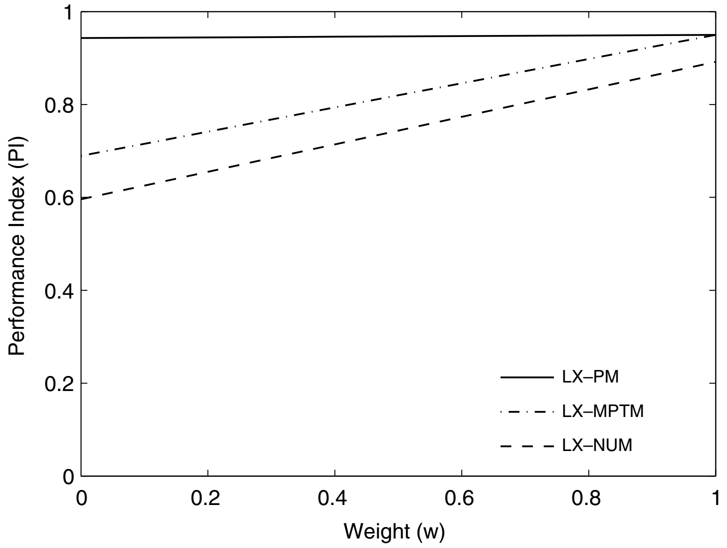

Fig. 3. Performance index of LX-PM, LX-MPTM, LX-NUM when $k_1 = w$ and $k_2 = k_3 = (\frac{1-w}{2})$.

$i = 1, 2, \ldots, N_p$

Sr^i = number of successful runs of ith problem
Tr^i = total number of runs of ith problem
At^i = average time used by an algorithm in obtaining the solution of ith problem
Mt^i = minimum of average time used by all the algorithms in obtaining the solution of ith problem
Af^i = average number of function evaluations of successful runs used by an algorithm in obtaining the solution of ith problem
Mf^i = minimum of average number of function evaluations of successful runs used all algorithms in obtaining the solution of ith problem
N_p = total number of problems analyzed

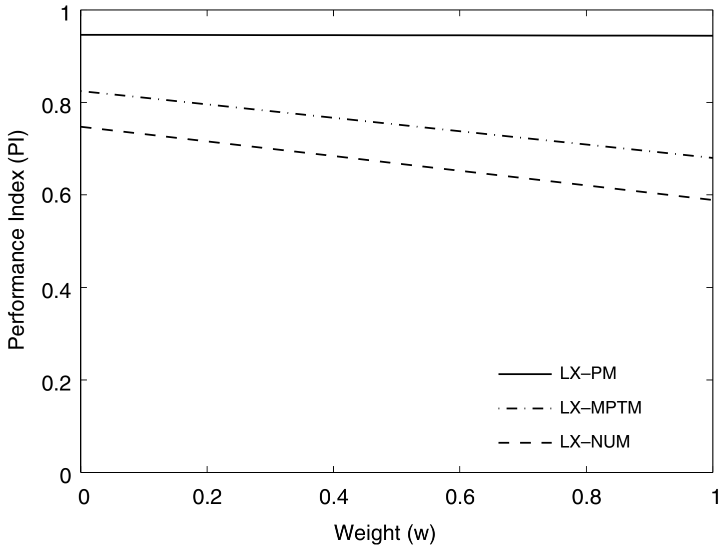

Fig. 4. Performance index of LX-PM, LX-MPTM, LX-NUM when $k_2 = w$ and $k_1 = k_3 = (\frac{1-w}{2})$.

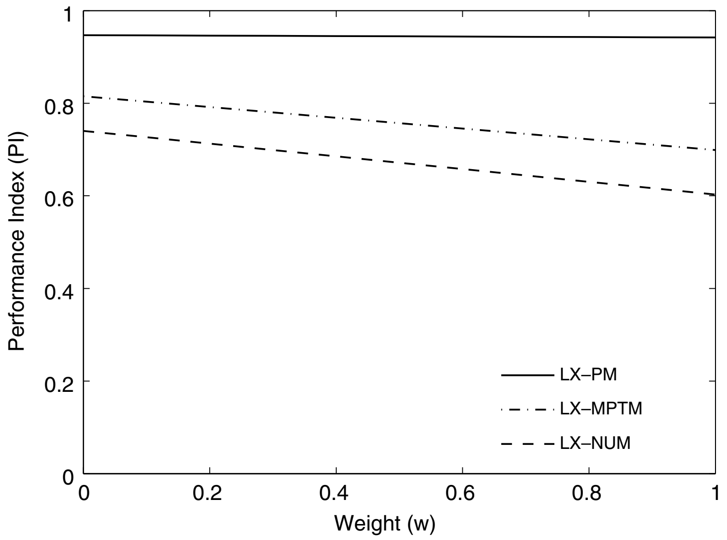

Fig. 5. Performance index of LX-PM, LX-MPTM, LX-NUM when $k_3 = w$ and $k_1 = k_2 = (\frac{1-w}{2})$.

k_1, k_2 and $k_3(k_1 + k_2 + k_3 = 1$ and $0 \leqslant k_1, k_2, k_3 \leqslant 1)$ are the weights assigned to percentage of success, execution time and average number of function evaluations of successful runs, respectively.

From above definition it is clear that PI is a function of k_1, k_2 and k_3. Since $k_1 + k_2 + k_3 = 1$, one of $k_i, i = 1, 2, 3$ could be eliminated to reduce the number of dependent variables from the expression of PI. But it is still difficult to analyze the behavior of PI, because the surface plots of PI for all GAs are overlapping and it is difficult to visualize them. So, we adopt the same methodology as given in Mohan and Nguyen [34], i.e. equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

(i) $k_1 = w,\ k_2 = k_3 = \frac{1-w}{2},\quad 0 \leqslant w \leqslant 1,$
(ii) $k_2 = w,\ k_1 = k_3 = \frac{1-w}{2},\quad 0 \leqslant w \leqslant 1,$
(iii) $k_3 = w,\ k_1 = k_2 = \frac{1-w}{2},\quad 0 \leqslant w \leqslant 1.$

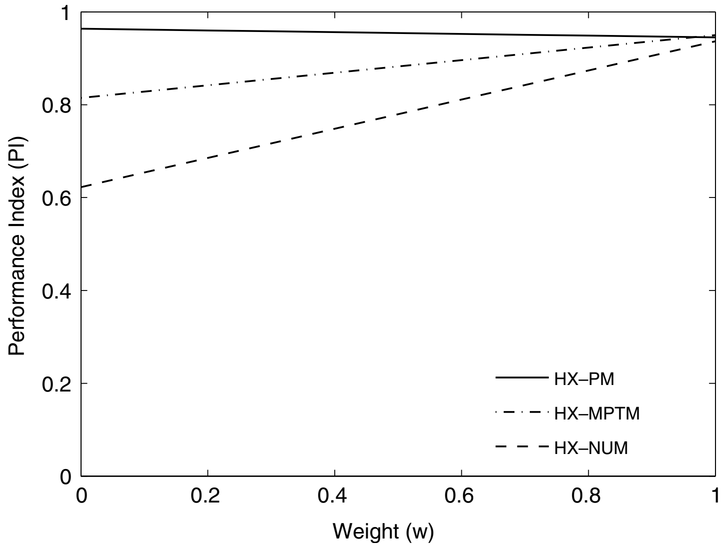

Fig. 6. Performance index of HX-PM, HX-MPTM, HX-NUM when $k_1 = w$ and $k_2 = k_3 = (\frac{1-w}{2})$.

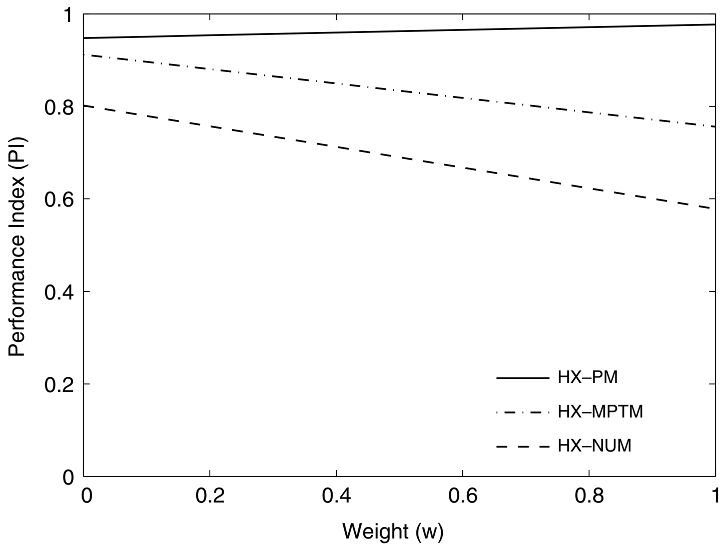

Fig. 7. Performance index of HX-PM, HX-MPTM, HX-NUM when $k_2 = w$ and $k_1 = k_3 = (\frac{1-w}{2})$.

The PI for the three GAs in Group 1 is shown in Figs. 3–5. It is evident that in all the three cases (as discussed above), the proposed mutation operator PM performs better than MPTM and NUM. Similar kind of results are observed for the three GAs in Group 2 as shown in Figs. 6–8.

To have an idea of the overall best GA we compare LX-PM with HX-PM. The reason is as follows. Since among the rest of the four GAs LX-MPTM is the best one as concluded by authors in [21] and from above discussion it is clear that LX-PM is even better than LX-MPTM. So it is sufficient to compare LX-PM and HX-PM to decide the best amongst the six GAs. The PI of LX-PM and HX-PM is shown in Figs. 9–11. It is clear that LX-PM beats HX-PM in all the three cases of PI. So we can conclude that LX-PM is the best amongst all the six GAs considered. However it is rather difficult to rank the remaining five GAs because the ranking varies according to the specified range of weights.

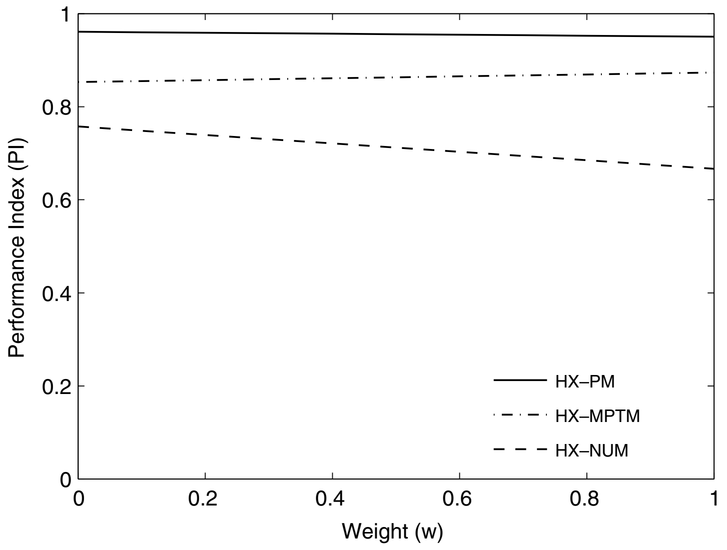

Fig. 8. Performance index of HX-PM, HX-MPTM, HX-NUM when $k_3 = w$ and $k_1 = k_2 = (\frac{1-w}{2})$.

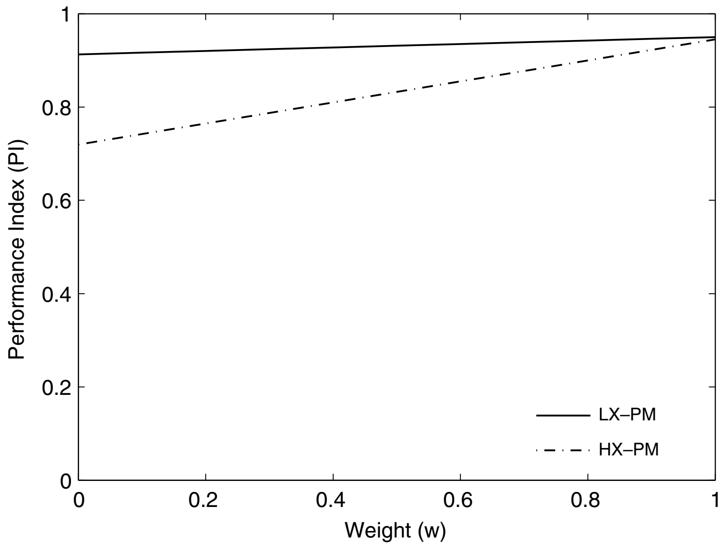

Fig. 9. Performance index of LX-PM, HX-PM when $k_1 = w$ and $k_2 = k_3 = (\frac{1-w}{2})$.

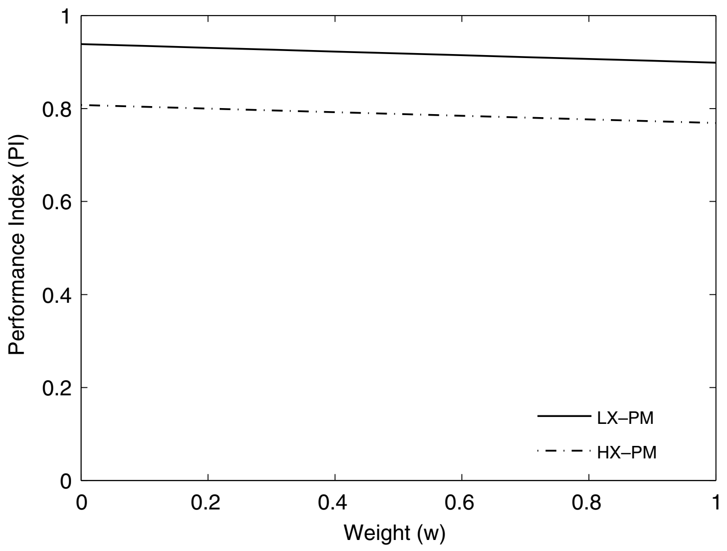

Fig. 10. Performance index of LX-PM, HX-PM when $k_2 = w$ and $k_1 = k_3 = (\frac{1-w}{2})$.

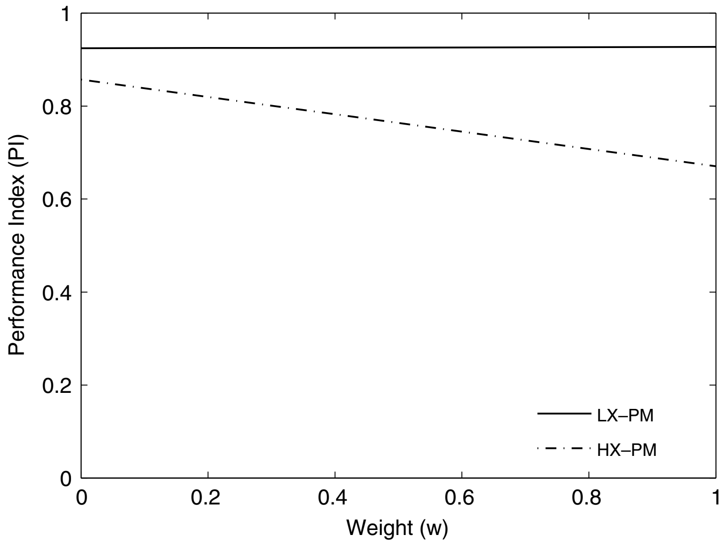

Fig. 11. Performance index of LX-PM, HX-PM when $k_3 = w$ and $k_1 = k_2 = (\frac{1-w}{2})$.

8. Conclusions

In this paper, a new mutation operator, called power mutation for real coded genetic algorithms is defined. It is based on the power distribution. The strength of power mutation is controlled by its index which gives rise to small (large) diversity as the value if the index is small (large). The efficacy of power mutation is established by evaluating its performance on the earlier set of 20 scalable test problems [21]. Using power mutation, two new generational RCGAs are designed by combining power mutation with the Laplace crossover and Heuristic crossover. Also the previously designed four GAs in [21] are used for comparison. The optimal parameter settings of all GAs are used with a view to make a fair comparison amongst the best of each of them.

For the sake of analysis, the six GAs are categorized into two groups. The three GAs in the first group use Laplace crossover in conjunction with the three varying mutation operators, whereas the three GAs in the second group use Heuristic crossover in conjunction with the three varying mutation operators.

Two kinds of analysis is performed. The basis of the first kind of analysis is the reliability (percentage of success), efficiency (average number of function evaluations and average computational time), accuracy (t-test results on mean of objective function value and standard deviation of objective function value). From the empirical study, it is concluded that amongst the GAs of the first group, the GA using Laplace crossover with power mutation emerged the best. Also, amongst the GAs of the second group, the GA using Heuristic crossover and power mutation is the best. Hence it may be concluded that power mutation outperforms all other mutation operators considered.

The second kind of analysis is carried by using a performance index, based on percentage of success, average number of function evaluations and average computational time. From the graphical results it is concluded that the power mutation outperforms the other mutation operators in GAs of both the groups. In the first group the GA using the Laplace crossover with power mutation is the best, whereas in the second group the GA using Heuristic crossover with power mutation is the best. Finally, on comparing these two winners it is concluded that the best GA is the one which uses Laplace crossover along with power mutation.

Acknowledgements

The authors are thankful to the anonymous referees for their valuable comments and suggestions. Also, the second author acknowledges Council for Scientific and Industrial Research (CSIR), New Delhi, India, for providing the financial support vide grant number 9/143 (439) 2003-EMR-I.

References

[1] T. Back, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation 1 (1) (1993) 1–23.

[2] K.A. De-Jong, An analysis of the behavior of a class of genetic adaptive systems. Doctoral Dissertation, University of Michigan, 1975.

[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.

[4] D.E. Goldberg, Real-coded genetic algorithms, virtual alphabets, and blocking, Complex Systems 5 (2) (1991) 139–168.

[5] C.Z. Janikow, Z. Michalewicz, An experimental comparison of binary and floating point representation in genetic algorithms, in: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, 1991, pp. 31–36.

[6] I. Ono, S. Kobayashi, A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover, in: T. Back (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1997, pp. 246–253.

[7] S. Tsutsui, M. Yamamura, T. Higuchi, Multi-parent recombination with simplex crossover in real-coded genetic algorithms, in: W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, R. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1 1999), 1999, pp. 657–664.

[8] H. Kita, I. Ono, S. Kobayashi, The multi-parent unimodal normal distribution crossover for real-coded genetic algorithms, in: Proceedings of the 1999 Congress on Evolutionary Computation, IEEE, vol. 2, 1999, pp. 1588–1595.

[9] K. Deb, A. Anand, D. Joshi, A computationally efficient evolutionary algorithm for real-parameter evolution, Evolutionary Computation Journal 10 (4) (2002) 371–395.

[10] A. Sinha, S. Tiwari, K. Deb, A population based steady state procedure for real parameter optimization. KANGAL Technical Report no. 2005004, 2005.

[11] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1992.

[12] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval schemata, in: D.L. Whitley (Ed.), Foundation of Genetic Algorithms II, Morgan Kaufmann, San Mateo, CA, 1993, pp. 187–202.

[13] N.J. Radcliffe, Equivalence class analysis of genetic algorithms, Complex Systems 2 (5) (1991) 183–205.

[14] H. Muhlebein, D. Schlierkamp-Voosen, Predictive models for breeder genetic algorithms in continuous parameter optimization, Evolutionary Computation 1 (1) (1993) 25–49.

[15] H.M. Voigt, H. Muhlenbein, D. Cvetkovic, Fuzzy recombination for the breeder genetic algorithms, in: L.J. Eshelmen (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1995, pp. 104–111.

[16] A.H. Wright, Genetic algorithms for real parameter optimization, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms I, Morgan Kaufmann, San Mateo, 1991, pp. 205–218.

[17] F. Herrera, M Lozano, Adaptation of genetic algorithm parameters based on fuzzy logic controllers, in: Genetic Algorithms and Soft Computing, 1996.

[18] F. Herrera, M. Lozano, J.L. Verdegay, Dynamic and heuristic fuzzy connectives based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms, International Journal Intelligent System 2 (1999) 1013–1041.

[19] K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex Systems 9 (1995) 115–148.

[20] S.H. Ling, F.H.F. Leung, An improved genetic algorithm with average-bound crossover and wavelet mutation operations, Soft Computing 11 (2007) 7–31.

[21] Kusum Deep, Manoj Thakur, A new crossover operator for real coded genetic algorithms, Applied Mathematics and Computations, accepted for publication, doi:10.1016/j.amc.2006.10.047.

[22] R.A.E. Makinen, J. Periaux, J. Toivanen, Multidisciplinary shape optimization in aerodynamics and electromagnetic using genetic algorithms, International Journal for Numerical Methods in Fluids 30 (2) (1999) 149–159.

[23] W.M. Spears, Crossover or mutation? in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 221–238.

[24] F. Herrera, M. Lozano, Two-Loop real coded genetic algorithms with adaptive control of mutation step sizes, Applied Intelligence 13 (2002) 187–204.

[25] J.E. Smith, T.C. Fogarty, Operators and parameter adaptation in genetic algorithms, Soft computing 1 (2) (1997) 81–87.

[26] N. Krasnogor, J.E. Smith, Emergence of profitable search strategies based on a simple inheritance mechanism, in: Proceedings of the International Conference on Genetic and Evolutionary Computations, Morgan Kaufmann, San Mateo, CA, 2001, pp. 432–439.

[27] J.E. Smith, Modelling GAs with self adapting mutation rates, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2001.

[28] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1996) 1–32.

[29] H. Muhelenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm 1, continuous parameter optimization, Evolutionary Computations 1 (1993) 25–49.

[30] H.M. Voigt, T. Anheyer, Modal mutations in evolutionary algorithms 1994, in: Proceedings of the First IEEE Conference on Evolutionary Computation, 1994, pp. 88–92.

[31] R. Hinterding, Gaussian mutation and self-adaption in numeric genetic algorithms, in: Proceedings of Second IEEE Conference Evolutionary Computation, IEEE Press, Piscataway, NJ, 1995, pp. 384–389.

[32] R. Hinterding, Z. Michalewicz, T.C. Peachey, Self-adaptive genetic algorithm for numeric functions, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Proceedings of the Fourth Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, vol. 1141, Springer, Berlin, Germany, 1996, pp. 420–429.

[33] A. Berry, P. Vamplew, PoD can mutate: a simple dynamic directed mutation approach for genetic algorithms, in: Proceedings of AISAT2004: International Conference on Artificial Intelligence in Science and Technology, Hobart, Tasmania, 2004.

[34] L. Temby, P. Vamplew, A. Berry, Accelerating Real-Valued Genetic Algorithms Using Mutation-with-Momentum, Lecture Notes in Computer Science, 3809, Springer-Verlag, 2005, pp. 1108–1111.

[35] C. Munteanu, C. Lazarescu, Improving mutation capabilities in a real-coded GA, in: Proceedings of GoIASP'99, Lecture Notes in Computer Science, 1596, Springer-Verlag, 1999, pp. 138–148.

[36] K. Deb, Muiti-objective Optimization Using Evolutionary Algorithms, Wiley, Chichester, 2001.

[37] Z. Michalewicz, T. Logan, S. Swaminathan, Evolutionary operators for continuous convex parameter space, in: A.V. Sebald, L.J. Fogel (Eds.), Proceeding of Third Annual Conference on Evolutionary Programming, World Scientific, River Edge, NJ, 1994, pp. 84–97.

[38] Z. Michalewicz, Genetic algorithms, numerical optimization and constraints, in: L.J. Eshelmen (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1995, pp. 151–158.

[39] Z. Michalewicz, D. Dasgupta, R.G. Le Riche, M. Schoenauer, Evolutionary algorithms for constrained engineering problems, Computers & Industrial Engineering Journal 30 (4) (1996) 851–870.

[40] K. Meittinen, M.M. Makela, J. Toivanen, Numerical comparison of some penalty based constraint handling techniques in genetic algorithms, Journal of Global Optimization 27 (2003) 427–446.

[41] H. Maaranen, K. Meittinen, M.M. Makela, Quasi random initial population for genetic algorithms, Computer and Mathematics with Applications 47 (2004) 1885–1895.

[42] Bharti, Controlled random search technique and their applications. Ph.D. Thesis. Department of Mathematics, University of Roorkee, Roorkee, India, 1994.

[43] C. Mohan, H.T. Nguyen, A controlled random search technique incorporating the simulating annealing concept for solving integer and mixed integer global optimization problems, Computational Optimization and Applications 14 (1999) 103–132.