Evolutionary Intelligence (2022) 15:2651–2666
https://doi.org/10.1007/s12065-021-00636-4

RESEARCH PAPER

Seeking a balance between population diversity and premature convergence for real-coded genetic algorithms with crossover operator

Fakhra Batool Naqvi[1] · **Muhammad Yousaf Shad**[1]

Received: 14 November 2020 / Revised: 17 March 2021 / Accepted: 29 June 2021 / Published online: 20 July 2021

Abstract
The major issue for optimization with genetic algorithms (GAs) is getting stuck on a local optimum or a low computation efficiency. In this research, we propose a new real-coded based crossover operator by using the Exponentiated Pareto distribution (EPX), which aims to preserve the two extremes. We used EPX with three the most reputed mutation operators: Makinen, Periaux and Toivanen mutation (MPTM), non uniform mutation (NUM) and power mutation (PM). The experimental results with eighteen well-known models depict that our proposed EPX operator performs better than the other competitive crossover operators. The comparison analysis is evaluated through mean, standard deviation and the performance index. Significance of EPX vs competitive is examined by performing the two-tailed t-test. Hence, the new crossover scheme appears to be significant as well as comparable to establish the crossing among parents for better offspring.

Keywords Genetic algorithms · Global optimization · Real-coded crossover operators · Exploration and exploitation

1 Introduction

For the global optima, several other techniques have been discussed in literature, e.g. simulated annealing (SA), differential evolution (DE), particle swarm optimization (PSO), ant-colony optimization (ACO), genetic algorithms (GAs) etc. [6] . In further studies, a detailed description about these algorithms have also been given [7–9]. Among these algorithms, GA has been found to be the most powerful algorithms to solve the optimization related problems [10]. In the early 1970s, professor Holland proposed the genetic algorithms (GAs) first time, see for examples [1–5]. It is a population based probabilistic approach, which searches the global optima for an optimistic problem.

There are several applications of GA, such as in automatic control, combinatorial optimization, production scheduling problems, optimization problems, planning and design, bio-engineering, system engineering, artificial intelligence and 6-6 parallel manipulators etc. [11–15, 55]. Also, many real life problems have been formulated as mathematical models to optimize their local objective functions, which further require their global optimum solution. This optimum value usually depends upon the decision variables that define the objective function. GA does not ensure the exact optimum solution, but it gives the optimal solution among the local optimum ones. Usually the real life problems are constrained optimization problems, but the current research deals with unconstrained optimization problems. It may appear as:

Minimize $g(y)$, with $g : R^n \rightarrow R$ where, $y \epsilon$ G. In the large search space, the G is denoted as a n-dimensional rectangular hypercube and R^n is identified by $c_i \leq y_i \leq d_i, i = 1, 2, 3, ..., n$. For evolutionary algorithms, the first step is to connect a bridge in the context of real situation problem and the problem solving space through evolutionary techniques. This step is defined as how the possible solutions are represented and stored in a computer language. In order to represent the candidates in the search space, a desired encoding scheme is adapted in which each of the chromosome is represented by the vector's length. Here the length of a vector is defined by the number of decision variables configuring the dimensions of search space. Most often these variables are represented in binary codes in the form of 0's and 1's.

✉ Fakhra Batool Naqvi
fakhrabatoolnaqvi@gmail.com

[1] Department of Statistics, Quaid-i-Azam University, Islamabad, Pakistan

The GA with binary coding problem is that it only maps the discrete values in the search set and this works well when an optimization problem has moderate decision variables. But in such cases, the accuracy of solution is compromised. As the accuracy of solution is directly linked to encoding length, so it results in excessive use of memory and computing which reduces the computation speed [16].

The idea of real encoding was emerged in early 90's, where a chromosome was interpreted in a vector of real coded GA [17–19]. The real-coded GA also uses three basic genetic operators i.e. selection, crossover and mutation. Although, it overcomes the problem of binary coded GA in a way that it requires continuous variables, but it may also faces the problem of premature convergence. This may happen due to the GA's inability to locally exploit the information regarding solutions in population.

GA has a major drawback that it gets stuck at local optimum because of its premature convergence. It is strongly linked with population diversity. For more selection pressure, there is less population diversity, which, as a result leads the GA to converge at the local optimum point. To maintain the balance between population diversity and selection pressure remains a major goal and it has been addressed by various studies [20, 21, 47, 48, 57] . We know that all the three operators affect the GA but crossover has major impact on it. It basically uses the information about the current population, directing the search in other regions of that search space. Hence, we can say that the exploration of GA depends on the crossover operator. Thus, it is essential to choose a suitable real-coded crossover operator to get more accurate results.

To overcome all the discussed issues with GAs, our study proposes an efficient crossover operator in order to come up with the solution for complex optimization problems. The current study is designed to have six sections. The Sect. 1 is about the introduction of GA as a tool to solve complex optimization problems. The Sect. 2 presents the existing real coded crossover operators. The Sect. 3 is about proposed real coded crossover operator, whereas the Sect. 4 is about an experimental setup. Section 5 discusses the results of the current study while the last Sect. 6 is all about concluding the current study findings.

2 Existing crossover operators

The performance of GA is highly affected by its operators, i.e. selection, crossover and mutation. Crossover is an important operator to maintain a balance between the two extremes, i.e. exploration and exploitation. In case of real coded GA, there is a list of operators that have been introduced in the literature. For example, Michalewicz [22] proposed a simple crossover related to genetics suggesting that how randomly selected genes from a parent pair are exchanged, result in production of the offspring. Radcliffe [23] proposed a flat crossover, which selects the genes from two parents by using the uniform distribution to produce an offspring. The search capabilities of this operator were further enhanced by extending the line and intermediate crossover operators, see for example, Muhlebein and Schlierkamp-Voosen [24]. Both of them are allowed the exploration within a pre-decided interval beyond the parents. Further, the above given ideas of Radcliffe, and Muhlebein and Schlierkamp-Voosen were further generalized by Eshelman and Schaffer [25]. They introduced the blend crossover operator with parameter 'α' which directs the exploration in the interval. The interval may be in between the parent's genes or on the either side of the parents. It also becomes the extended intermediate crossover for $\alpha = 0.25$. In 1991, Wright [18] introduced the heuristic crossover (HX) based on the fitness values of parents. Only one offspring is produced by mating of two parents and it is also biased in the favor of relatively better parent.

A lot of approaches to generate offspring through arithmetical crossovers were suggested by Michalewicz [26]. For examples, one of them is to produced offspring within the genes interval of both parents. Another one is to generate one offspring using uniform distribution between the genes and using the means of parents to generate the second one. The idea of fuzzy recombination operators, which was used with heuristic crossover as heuristic fuzzy connective based crossovers, see for example, [27, 28]. These crossovers successfully maintain the population's diversity with enhanced convergence speed. A simplex multi-parent crossover proposed by Tsutsui [29] produces the offspring from the simplex formed by the parent solutions. An improved version of Genetic Diversity Evolutionary Algorithm (GeDEA) called GeDEA-II is proposed which features a novel crossover operator, the Simplex-Crossover (SPX), and a novel mutation operator, the Shrink-Mutation [56] .The simulated binary crossover (SBX) aims to simulate binary-crossovers making it useful for continuous search space [30]. A Gaussian distribution based crossover for real-coded GA was proposed by Tutkun [31]. Laplace crossover (LX) was suggested by Deep and Thakur [32], which produces offspring based on the Laplace distribution. Logistic crossover (LogX) was proposed to produce more near optimal results based on Logistic distribution [57] .

Some other crossover operators have also been proposed with multiple descendants. Producing three offspring from two parents using a linear crossover operator [18]. A unimodal normally distributed crossover operator (UNDX) introduced by Ono and Kobayashi [33] where three parents participate to produce two or more offspring. Later on, its performance is enhanced by adding the uniform crossover (UX) [34]. Another multi-parent crossover operator called

the Parent centric crossover (PCX) was proposed by Deb et al. [35]. It was further modified by Sinha et al. [36]. In the average bound crossover, four offspring are produced and the parents are then replaced by the two best offspring [37]. The hybrid crossover operator produces a number of offspring using different crossover operators. Crossover operators are classified as mean centric or parent centric if they generate offspring near the centroid of parent or near the parents. A detailed analysis was done by Herrera et al. [38]. The simplex crossover, blend crossover and the unimodal normal distribution crossover are the mean centric operators, whereas, LX, SBX and fuzzy recombination are parent centric approaches.

The differential evolution crossover (DEX) has been proposed as a multi-parent crossover operator to avoid premature convergence. As a part of improved class of RCGAs it uses successful parent strategy which provides a successful alternative to parent selection during DEX process [43]. A new hybrid strategy was applied by combining the SBX and simplex crossover to effectively optimize a problem [44]. Another recent variant of RCGA is the improved real coded genetic algorithm (IRCGA) which uses SBX to improve the convergence speed and solution quality for dynamic economic dispatch [45]. Another idea used a multi-parent crossover operator with a diversity operator instead of a mutation operator to solve constrained optimization problems as well as engineering problems [46]. A new adaptive genetic algorithm has been proposed to maintain the balance between exploration and exploitation. It used the arithmetic crossover operator in a new adaptive environment [47]. For the optimization of multi-modal test problems for real coded genetic algorithm, a novel parent centric crossover operator is proposed based on a log-logistic probability distribution. The main aim of fisk crossover (FX) is trade-off between selection pressure and population diversity [48]. An improved RCGA is defined by using a new heuristic normal distribution crossover (HNDX) which aim to direct the crossover to the optimal crossover direction [49]. Direction based crossover operators have also been proposed which direct the crossover search direction to be consistent with the optimal crossover direction [50–53]. Another conditionally breeding RCGA is performed by difference degree between individuals instead of using the crossover and mutation probability. It aimed to improve the ability of GA to converge to the near optimal solution [54].

3 Real-coded crossover operators: used in this study

The operators that have been used in the current study are:

3.1 Laplace crossover (LX)

Based on Laplace distribution, Deep and Thakur [32] proposed a self-adaptive parent centric crossover operator. The Laplace distribution function is as follows.

$$F(y) = \begin{cases} \frac{1}{2}\exp(\frac{|y-l|}{m}), & y \le l \\ 1 - \frac{1}{2}\exp(-\frac{|y-l|}{m}), & y > l \end{cases} \tag{1}$$

Following the same steps as mentioned earlier, a parameter $\}\beta_i$' is created using following formula:

$$\beta_i = \begin{cases} l - blog_e(u_i), & u_i \le 0.5 \\ l + blog_e(u_i), & u_i > 0.5 \end{cases} \tag{2}$$

Offspring are generated using following equations:

$$\begin{cases} \xi_i = y_i + \beta_i|y_i - z_i| \\ \eta_i = z_i + \beta_i|y_i - z_i| \end{cases} \tag{3}$$

The authors suggested to assign zero value to the location parameter of this distribution.

3.2 Simulated binary crossover (SBX)

Simulated binary crossover is among the most commonly known crossover operator which was proposed by Deb and Agarwal [30]. The cumulative distribution function (CDF) is as follows:

$$F(y) = 0.5y^{m_c+1} \tag{4}$$

The parameter 'β_i' is generated using the following equations:

$$\beta_i = \begin{cases} (2u_i)^{\frac{1}{m_c+1}}, & u_i \le 0.5 \\ \frac{1}{(2(1-u_i))^{\frac{1}{m_c+1}}}, & u_i > 0.5 \end{cases} \tag{5}$$

where, $m_c \epsilon (0, \infty)$ is the distribution index for SBX. Two offspring $\theta = (\theta_1, \theta_2, ..., \theta_n)$ and $\eta = (\eta_1, \eta_2, ..., \eta_n)$ are generated from a pair of parents $y = (y_1, y_2, ..., y_n)$ and $z = (z_1, z_2, ..., z_n)$ as follows:

$$\begin{cases} \theta_i = 0.5((y_i + z_i) - \beta_i|y_i - z_i|) \\ \eta_i = 0.5((y_i + z_i) - \beta_i|y_i - z_i|) \end{cases} \tag{6}$$

3.3 Heuristic crossover (HX)

This is the most common and elderly used crossover is Heuristic crossover which was proposed by Wright [18]. It aims to solve the constrained and unconstrained optimization problems. It works differently from other crossover operators as it produces a single offspring from a parent pair. The steps to generate offspring $\xi = (\xi_1, \xi_2, ..., \xi_n)$ are as follows:

1. Generate a random number u_i between 0 and 1.
2. Find an offspring using following formula:

$$\xi_i = (z_i - y_i)u_i + z_i \tag{7}$$

and observe the fitness of the parent z, which is not worse than the parent y. Produced offspring will be biased in the direction of relatively fit parent.

In case the produced offspring $\xi \leq y_i^l$ or $\xi \geq y_i^u$, then a new offspring is generated using step 2. The produced offspring will be assigned a random value if HX fails to produce an offspring within the constraints even after n attempts as:

$$\xi_i = y_i^l + (y_i^u - y_i^l)u_i, \tag{8}$$

where y_i^l and y_i^u are the lower and upper limits of i^{th} parent y.

3.4 Logistic crossover (LogX)

One recently proposed crossover operator based on the Logistic distribution [57]. The CDF of Logistic distribution is given as:

$$F(x) = \frac{1}{(1 + \exp^{-(\frac{(x-\mu)}{s})})} \tag{9}$$

From a pair of parents $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$, two offspring are generated as $\xi_i = (\xi_1, \xi_2, ..., \xi_n)$ and $\eta_i = (\eta_1, \eta_2, ..., \eta_n)$ using the following steps:

1. Generate a random number u between 0 and 1.
2. The parameter β_i is created by inverting the CDF of Logistic distribution.

$$\beta_i = \mu - sLog(\frac{1-u}{u}) \tag{10}$$

3. For i=1,2,...,n, the offspring are given by the equation:

$$\eta_i = 0.5[(x + y) + \beta_i|x - y|)] \tag{11}$$

$$\xi_i = 0.5[(x + y) - \beta_i|x - y|)] \tag{12}$$

3.5 Makinen, Periaux and Toivanen mutation (MPTM)

This mutation operator is suggested by Makinen et al. [39]. It was proposed to solve multidisciplinary shape optimization problems in aerodynamics and electromagnetic. It is also used to solve constrained optimization problems [40]. The mutated individual $\zeta = (\zeta_1, \zeta_2, ..., \zeta_n)$ is created from an individual $\eta = (\eta_1, \eta_2, ..., \eta_n)$ as follows:

1. A uniform random number, say u, is generated between 0 and 1.
2. Find

$$k = \frac{x_i - (x_i)^l}{(x_i)^u - x_i}, \tag{13}$$

where x_i^l and x_i^u are the lower and upper bounds of ith decision variable respectively.

3. A parameter k_i is created using following formula:

$$k_i = \begin{cases} k_i - k_i(\frac{k_i - r_i}{k_i})^b, & if \;\; r_i < k_i \\ k_i, & if \;\; r_i = k_i \\ k_i + (1 - k_i)(\frac{r_i - k_i}{1 - k_i})^b, & if \;\; r_i > k_i \end{cases} \tag{14}$$

where ' b' is the index of mutation.

4. The muted solution will be created as follows:

$$\zeta_i = (1 - k_i)x_i^l + k_i x_i^u \tag{15}$$

3.6 Non-uniform mutation (NUM)

It is one of the most common and widely used mutation operator proposed by [41, 42]. In this type of mutation as the number of generation increases, there is decrease in the strength of mutation. In other words, the NUM operator uniformly searches for the space in the initial generations, but in the later generations it searches for the space locally. The steps to create the muted solution are:

1. Create a uniformly distributed random number between 0 and 1.
2. The muted solution will be:

$$\zeta_i = \begin{cases} \epsilon_i + (x_i^u - \epsilon_i)(1 - u_i^{(1-(t/T))})^b, & if \;\; r_i \leq 0.5 \\ \epsilon_i - (\epsilon_i - x_i^l)(1 - u_i^{(1-(t/T))})^b, & otherwise \end{cases} \tag{16}$$

where *'b'* is the parameter of mutation operator which determines the strength of mutation. *'T'* stands for the maximum number of generations whereas *'t'* represents the current generation number.

3.7 Power mutation (PM)

One of the most frequently used operator that uses Power distribution was originally proposed by Deep and Thakur [32]. Its density function is as follows:

$$f(y) = \lambda y^{\lambda-1}, \qquad 0 \le y \le 1 \tag{17}$$

with the following distribution function:

$$F(y) = y^{\lambda}, \qquad 0 \le y \le 1 \tag{18}$$

where 'λ' is the index of the distribution. Following steps are used to find the muted solution.

1. A random number 'r' is created between 0 and 1.
2. Create a random number 's_i' which follows the Power distribution as:

$$s_i = (r_i)^{1/\lambda} \tag{19}$$

3. Create the muted solution using the following equation:

$$\zeta_i = \begin{cases} \epsilon_i - s_i(\epsilon_i - y_i^l), & if \ \frac{y_i - y_i^l}{y_i^u - y_i} < r_i \\ \epsilon_i + s_i(y_i^u - \epsilon_i), & if \ \frac{y_i - y_i^l}{y_i^u - y_i} \ge r_i \end{cases} \tag{20}$$

where r_i is the uniform random number between 0 and 1 and y_i^l and y_i^u are the lower and upper bounds of i^{th} decision variable respectively.

4 The proposed real coded crossover operator based on Exponentiated Pareto distribution

The proposed Exponentiated Pareto crossover (EPX) operator is used in conjunction with the Makinen, Periaux and Toivanen mutation (MPTM). This modification give rise to a new crossover operator EPX-MPTM to improve the performance of existing crossover operators. The density function of Exponentiated pareto distribution is given as follows:

$$f(t) = \mu\theta(1-(1+t)^{-\mu})^{\theta-1}(1+t)^{-(\mu+1)} \tag{21}$$

where $\mu, \theta > 0$ are the location and scale parameters respectively. The CDF of Exponentiated Pareto distribution is given as:

$$F(t) = (1-(1+t)^{-\mu})^{\theta} \tag{22}$$

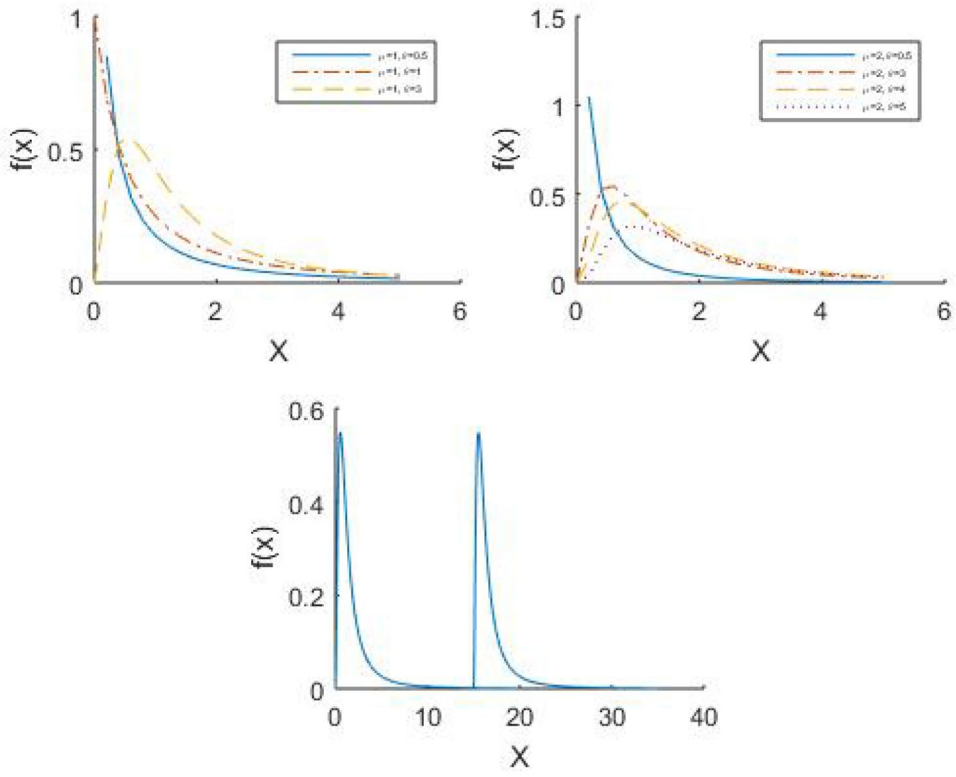


Fig. 1 Exponentiated Pareto distribution for different values of parameters, Distribution of offspring

Table 1 Summary of operators used in the study

Operators used				
Algorithm name	Selection	Crossover	Mutation	Elitism
EPX-MPTM	Tournament	EPX	MPTM	Yes
LX-MPTM	Tournament	LX	MPTM	Yes
SBX-MPTM	Tournament	SBX	MPTM	Yes
HX-MPTM	Tournament	HX	MPTM	Yes
LogX-MPTM	Tournament	LogX	MPTM	Yes
EPX-NUM	Tournament	EPX	NUM	Yes
LX-NUM	Tournament	LX	NUM	Yes
SBX-NUM	Tournament	SBX	NUM	Yes
HX-NUM	Tournament	HX	NUM	Yes
LogX-NUM	Tournament	LogX	NUM	Yes
EPX-PM	Tournament	EPX	PM	Yes
LX-PM	Tournament	LX	PM	Yes
SBX-PM	Tournament	SBX	PM	Yes
HX-PM	Tournament	HX	PM	Yes
LogX-PM	Tournament	LogX	PM	Yes

Table 2 Parameter settings used for GA

Parameter	Settings
Representation	Real
Population size	300
Crossover schemes	EPX, SBX, LX, HX and LogX
Crossover probability	75%
Mutation operator	MPTM, NUM, PM
Mutation rate	5%
Maximum generation	1000
Number of dimensions	30

From parents $k = (k_1, k_2, ..., k_n)$ and $l = (l_1, l_2, ..., l_n)$, two offspring are generated as $\xi_i = (\xi_1, \xi_2, ..., \xi_n)$ and $\eta_i = (\eta_1, \eta_2, ..., \eta_n)$ using the steps given below:

1. Generate a random number 'z', where $z\epsilon$ (0,1).
2. The parameter 'α_i' is created by inverting the CDF of Exponentiated pareto Distribution i.e., by equating the area under the curve from $-\infty$ to α_i to the randomly generated number z as follows:

$$\begin{aligned} F(t) &= (1-(1+t)^{-\mu})^{\theta} \\ z &= (1-(1+\alpha_i)^{-\mu})^{\theta} \\ z^{1/\theta} &= 1-(1+\alpha_i)^{-\mu} \\ 1-z^{1/\theta} &= (1+\alpha_i)^{-\mu} \\ (1-z^{1/\theta})^{(-1/\mu)} &= 1+\alpha_i \end{aligned} \tag{23}$$

$$\alpha_i = ((1-z^{1/\theta})^{(-1/\mu)}) - 1 \tag{24}$$

3. For i=1,2,...,n, the offspring are given by the equation:

$$\begin{cases} \eta_i = 0.5[(k+l) + \alpha_i|k-l|)] \\ \xi_i = 0.5[(k+l) - \alpha_i|k-l|)] \end{cases} \tag{25}$$

The shape of Exponentiated Pareto distribution is shown in the upper left subplot of Fig. 1. The distribution becomes symmetric for larger values of the parameter 'θ'. The upper right subplot of Fig. 1 clearly depicts that for a fixed value of 'μ', a smaller value of parameter 'θ' generates near the parent offspring while the larger value of this parameter generates far off parents. As we know genetic algorithm is a population based algorithm and the main purpose of any operator (i.e. selection, crossover and mutation) is to keep balance between selection pressure and population diversity. The long tail of the Exponentiated Pareto distribution shows high population diversity which reduces the selection pressure and vice versa for the high peak. But the Exponentiated Pareto distribution maintains a better trade-off between the population diversity and selection pressure (as seen in the upper left subplot of Fig. 1).

The third subplot shows the distribution of offspring. It shows increase in probability of generating offspring away from parent. Meanwhile it also depicts that the length of interval, in which offspring are generated, increases with an increase in 'θ' (the spread parameter). EPX is very explorative and its exploration range also increases with an increase in 'θ' (shown by long tail in graph). In this way, the search power of the proposed operator in terms of probability of generating the offspring is high.

5 The experimental setup

In this section, experimental test is carried out with its eighteen benchmark functions in order to validate the proposed algorithm. It is aimed to make its comparison with three other optimization algorithms.

5.1 Mathematical test functions

Eighteen benchmark test problems have been selected to test the performance of proposed crossover operator. These problems have different difficulty level and multi-modality. The problems with their essential information are summarized in appendix.

5.2 Algorithms for comparison

For performance evaluation of the proposed algorithm, the optimal fitness values of EPX-MPTM are compared with four other evolutionary algorithms. These are LX, SBX,

Table 3 Mean, standard deviation and number of succesful runs for 1000 simulated results with 30 independent runs

Problem Name	EPX-MPTM		LX-MPTM		SBX-MPTM		HX-MPTM	
	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs
Cigar	3.27E+01 (7.19E+01)	6	6.22E+02 (1.44E+03)	0	4.02E+01 (9.54E+01)	8	5.58E+02 (1.17E+03)	2
New	4.70E-06 (1.27E-05)	30	1.87E-04 (3.85E-04)	30	2.33E-05 (4.29E-05)	30	2.37E+00 (1.11E+01)	30
Greiwank	3.51E-02 (5.03E-02)	22	2.43E-01 (3.08E-01)	12	4.18E-02 (6.57E-02)	22	1.93E-01 (2.95E-01)	2
Cosine Mixture	-3.00E+00 (8.88E-06)	30	-3.00E+00 (7.63E-05)	30	-3.00E+00 (1.62E-05)	30	-2.9732 (1.04E-01)	30
Brown	4.18E-07 (1.13E-06)	30	3.42E-05 (4.42E-05)	30	2.91E-06 (9.08E-06)	30	1.28E-05 (2.04E-05)	30
Generalized1	4.09E-08 (8.67E-08)	30	7.04E-06 (6.67E-06)	30	7.97E-07 (1.82E-06)	30	7.80E-03 (3.01E-02)	30
Generalized2	7.67E-05 (1.95E-04)	30	3.00E-03 (7.60E-03)	30	2.20E-03 (4.60E-03)	30	8.60E-03 (2.33E-02)	30
Sphere	7.76E-06 (1.55E-05)	30	2.81E-04 (4.93E-04)	30	9.68E-06 (2.06E-05)	30	4.19E-05 (6.47E-05)	30
New 13	2.03E-06 (2.57E-06)	30	6.54E-04 (1.10E-03)	30	5.60E-06 (7.02E-06)	30	2.53E+00 (9.80E+00)	30
Hyper Ellipsoid	4.76E-05 (9.88E-05)	30	2.80E-03 (7.10E-03)	30	2.45E-04 (5.58E-04)	30	1.10E-03 (2.40E-03)	30
LevyMount1	1.65E-06 (3.27E-06)	30	2.51E-05 (3.70E-05)	30	1.80E-06 (6.85E-06)	30	4.88E-02 (1.34E-01)	30
LevyMount2	7.79E-06 (1.23E-05)	30	7.17E-05 (1.80E-04)	30	6.30E-03 (9.10E-03)	30	6.70E-03 (2.57E-02)	30
Ellipsoidal	5.31E+00 (1.48E+01)	6	5.03E+02 (1.23E+02)	0	8.68E+00 (1.49E+01)	2	1.34E+03 (2.97E+02)	0
Dejong's	1.28E+01 (8.46E+00)	0	1.40E+01 (7.55E+00)	0	1.48E+01 (6.80E+00)	0	1.37E+01 (8.66E+00)	0
Rosenbrock	1.84E-01 (3.83E-01)	20	6.58E+00 (1.13E+01)	12	1.57E+00 (6.68E+00)	22	1.98E-01 (3.89E-01)	10
step	6.24E-09 (9.58E-09)	30	1.57E-02 (2.80E-02)	26	9.92E-07 (2.07E-06)	30	1.01E-02 (1.21E-02)	30
Rastrigin	1.85E+00 (4.00E+00)	24	2.14E+01 (2.83E+01)	14	3.97E+00 (3.94E+00)	10	9.48E+01 (6.31E+01)	6
Neumair	1.80E+01 (3.12E+00)	30	2.69E+01 (1.38E+01)	30	2.18E+01 (1.48E+01)	30	3.47E+01 (6.25E+01)	28

HX and LogX in conjunction with three mutation operators MPTM, NUM and PM.

5.3 Settings for comparison

In order to maintain uniformity of the testing environment, the population size for all the cases is made ten times to the number of decision variables. There are thirty independent runs which are made on various initial populations using each of the algorithm. All the GAs use tournament selection as a selection criteria. Elitism is applied with size one i.e., the best individuals are preserved in the current generation. The value of 'a' for LX is set to be zero and for HX, 'k' is 4. A maximum number of 1000 generations is the stopping criteria for all the algorithms. The values of parameters are fixed during a complete run. The performance of the

Table 4 Mean, standard deviation and number of succesful runs for 1000 simulated results with 30 independent runs

Problem name	EPX-NUM		LX-NUM		SBX-NUM		HX-NUM	
	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs
Cigar	1.27E+01 (1.43E+01)	0	4.87E+04 (7.71E+04)	0	6.10E+01 (1.07E+02)	12	8.25E+03 (1.89E+04)	12
New	1.43E-06 (2.16E-06)	30	3.94E+00 (3.57E+00)	4	2.39E-06 (5.60E-06)	30	3.69E-01 (9.76E-01)	26
Greiwank	2.70E-02 (3.01E-02)	24	2.18E-02 (2.37E-02)	24	1.03E-01 (8.93E-02)	20	2.37E-01 (4.07E-01)	16
Cosine Mixture	-3.00E+00 (1.45E-06)	0	-1.51E+00 (3.97E-01)	30	-3.00E+00 (2.13E-05)	30	-2.2683 (1.10E+00)	30
Brown	4.14E-11 (4.51E-11)	30	4.03E-07 (5.67E-07)	30	5.86E-06 (1.17E-05)	30	4.11E-04 (6.49E-04)	30
Generalized1	7.89E-07 (1.50E-06)	30	3.47E-01 (3.54E-01)	8	3.20E-05 (1.17E-04)	30	5.21E-01 (7.71E-01)	24
Generalized2	1.70E-03 (3.90E-03)	30	6.85E-02 (9.24E-02)	18	9.50E-03 (1.75E-02)	28	1.42E-01 (2.46E-01)	16
Sphere	2.09E-10 (3.44E-10)	30	1.64E-08 (3.88E-08)	30	1.06E-05 (1.67E-05)	30	2.23E-04 (5.97E-04)	30
New 13	8.45E-04 (2.40E-03)	30	1.03E+02 (5.08E+01)	0	3.12E-01 (6.75E-01)	24	8.17E+01 (4.49E+01)	0
Hyper Ellipsoid	8.18E-05 (1.61E-04)	30	3.00E-01 (2.54E-01)	8	9.35E-04 (3.10E-03)	30	3.98E-02 (1.34E-01)	30
LevyMount1	4.41E-07 (4.17E-07)	30	4.86E-02 (5.38E-02)	16	6.97E-06 (1.38E-05)	30	1.33E-01 (2.04E-01)	26
LevyMount2	1.10E-03 (3.00E-03)	30	1.67E-01 (3.02E-01)	22	5.10E-03 (6.80E-03)	30	3.06E-02 (1.01E-01)	28
Ellipsoidal	1.57E-07 (1.92E-07)	30	6.92E-04 (8.55E-04)	30	4.09E-01 (1.39E+00)	24	1.27E+03 (3.03E+02)	0
Dejong's	1.41E+01 (7.00E+00)	0	1.62E+01 (1.03E+01)	0	1.46E+01 (9.75E+00)	0	1.63E+01 (6.98E+00)	0
Rosenbrock	4.32E+01 (2.95E+01)	2	1.15E+03 (2.74E+03)	0	6.72E+01 (3.38E+01)	0	1.27E+01 (3.55E+00)	18
step	6.74E-32 (2.56E-31)	30	1.55E-11 (3.74E-11)	30	5.18E-12 (5.89E-12)	30	1.43E-32 (5.54E-32)	30
Rastrigin	8.43E+00 (2.22E+00)	0	7.46E+01 (1.55E+01)	0	5.60E+00 (4.60E+00)	0	1.46E+02 (1.87E+01)	0
Neumair	1.43E+03 (1.47E+03)	6	2.89E+02 (2.68E+02)	20	5.83E+02 (2.64E+02)	2	5.23E+01 (1.40E+02)	28

proposed algorithm has been evaluated on each test function based on the statistics from 30 independent runs. A run is said to be successful if the objective function value varies not more than 5% of its optimal value in the run. In this way, performance index has been computed.

6 Results and discussion

In this section, comparison of the proposed operators EPX-MPTM, EPX-NUM and EPX-PM is performed with other crossover operators in groups of three i.e., (HX-MPTM, LX-MPTM, SBX-MPTM), (HX-NUM, LX-NUM, SBX-NUM) and (HX-PM, LX-PM, SBX-PM). The mutation operator is same in each of these groups to maintain

Table 5 Mean, standard deviation and number of succesful runs for 1000 simulated results with 30 independent runs

Problem name	EPX-PM		LX-PM		SBX-PM		HX-PM	
	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs
Cigar	6.05E+04 (1.83E+05)	0	9.29E+06 (6.35E+06)	0	8.31E+04 (8.86E+04)	0	4.07E+07 (1.96E+07)	0
New	2.40E-03 (4.30E-03)	30	6.44E+01 (3.43E+01)	0	2.47E-02 (3.57E-02)	28	1.17E+02 (5.13E+01)	0
Greiwank	3.15E-01 (3.41E-01)	0	1.54E+01 (7.77E+00)	0	3.15E-01 (2.62E-01)	8	3.39E+01 (9.12E+00)	0
Cosine Mixture	-2.99E+00 (6.00E-03)	30	-2.84E+00 (1.94E-01)	30	-3.00E+00 (5.90E-03)	30	-9.62E-01 (3.46E-01)	30
Brown	7.88E-04 (1.60E-03)	30	2.34E-01 (1.32E-01)	24	8.30E-03 (6.80E-03)	30	1.28E+00 (4.32E-01)	4
Generalized1	1.39E-05 (1.92E-05)	30	4.37E-02 (5.48E-02)	26	6.09E-05 (1.24E-04)	30	5.64E-01 (9.02E-01)	10
Generalized2	3.30E-03 (4.30E-03)	30	4.83E-01 (3.62E-01)	4	3.60E-03 (6.30E-03)	30	9.71E-01 (4.09E-01)	0
Sphere	7.87E-04 (1.20E-03)	30	3.08E+00 (1.98E+00)	0	5.20E-03 (1.03E-02)	30	1.08E+01 (3.13E+00)	0
New 13	4.92E-01 (5.24E-01)	10	6.44E+01 (3.43E+01)	0	5.21E-01 (5.75E-01)	18	1.17E+02 (5.13E+01)	0
Hyper Ellipsoid	1.06E-01 (2.37E-01)	22	4.35E+01 (4.62E+01)	0	3.78E-01 (6.53E-01)	10	1.34E+02 (4.56E+01)	0
LevyMount1	2.89E-05 (4.89E-05)	30	1.14E-01 (8.70E-02)	8	6.38E-04 (6.99E-04)	30	3.98E-01 (2.18E-01)	0
LevyMount2	1.90E-03 (4.00E-03)	30	2.63E-01 (2.76E-01)	0	5.60E-03 (7.20E-03)	30	1.02E+00 (5.86E-01)	2
Ellipsoidal	5.18E-01 (9.47E-01)	8	4.38E+02 (1.47E+02)	0	8.12E+00 (1.18E+01)	0	1.48E+03 (3.73E+02)	0
Dejong's	1.25E+01 (8.82E+00)	0	1.56E+01 (8.73E+00)	0	1.46E+01 (1.02E+01)	0	1.26E+02 (6.65E+01)	0
Rosenbrock	3.30E+02 (3.55E+02)	0	9.80E+04 (2.34E+05)	0	3.16E+02 (1.76E+02)	0	1.09E+06 (1.19E+06)	14
step	1.70E-13 (6.60E-13)	30	0.00E+00 (0.00E+00)	30	1.08E-11 (4.18E-11)	30	2.23E+03 (8.64E+03)	28
Rastrigin	7.21E+00 (3.20E+00)	0	5.37E+01 (2.85E+01)	0	4.22E+00 (1.83E+00)	0	1.44E+02 (1.67E+01)	0
Neumair	2.44E+03 (2.42E+03)	0	1.28E+05 (8.52E+04)	4	3.19E+03 (1.76E+03)	0	2.75E+05 (1.01E+05)	18

uniformity of the comparison process. Results are analyzed in three different ways. EPX operator performed better than other crossover operators in most of the test problems. It is also compared with the LogX operator using each of three mutation operators . Results showed that the proposed operator is more efficient in function optimization as compared to the other crossover operators. The Table 1 shows the summary of operators used in the study (Table 1. The parameter settings used for the GA are shown in Table 2.

Table 6 Mean, standard deviation and number of succesful runs of LogX and EPX for 1000 simulated results with 30 independent runs

Problem name	EPX-MPTM		LogX-MPTM		EPX-NUM		LogX-NUM	
	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs	Mean (S.D)	Successful runs
Cigar	3.27E+01 (7.19E+01)	6	4.30E+02 (3.53E+02)	0	1.27E+01 (1.43E+01)	0	2.04E+05 (1.25E+05)	0
New	4.70E-06 (1.27E-05)	22	1.20E-03 (2.00E-03)	30	1.43E-06 (2.16E-06)	30	1.17E+01 (7.68E+00)	0
Greiwank	3.51E-02 (5.03E-02)	30	2.07E-01 (2.14E-01)	8	2.70E-02 (3.01E-02)	24	1.22E+00 (1.73E-01)	0
Cosine Mixture	-3.00E+00 (8.88E-06)	30	-3.00E+00 (3.66E-04)	30	-3.00E+00 (1.45E-06)	0	-2.79E+00 (1.62E-01)	30
Brown	4.18E-07 (1.13E-06)	30	4.32E-05 (5.14E-05)	30	4.14E-11 (4.51E-11)	30	5.40E-02 (2.95E-02)	14
Generalized1	4.09E-08 (8.67E-08)	30	3.65E-05 (4.20E-05)	30	7.89E-07 (1.50E-06)	30	1.00E-02 (2.64E-02)	28
Generalized2	7.67E-05 (1.95E-04)	30	3.56E-05 (5.55E-05)	30	1.70E-03 (3.90E-03)	30	6.29E-02 (5.86E-02)	18
Sphere	7.76E-06 (1.55E-05)	30	2.69E-04 (2.25E-04)	30	2.09E-10 (3.44E-10)	30	7.39E-02 (4.95E-02)	14
New 13	2.03E-06 (2.57E-06)	30	7.38E-04 (8.59E-04)	30	8.45E-04 (2.40E-03)	30	2.74E-01 (3.74E-01)	16
Hyper Ellipsoid	4.76E-05 (9.88E-05)	30	6.80E-03 (1.31E-02)	28	8.18E-05 (1.61E-04)	30	1.06E+00 (7.03E-01)	0
LevyMount1	1.65E-06 (3.27E-06)	30	4.27E-05 (6.37E-05)	30	4.41E-07 (4.17E-07)	30	9.60E-03 (2.66E-02)	28
LevyMount2	7.79E-06 (1.23E-05)	30	4.55E-05 (4.02E-05)	30	1.10E-03 (3.00E-03)	30	6.45E-02 (4.99E-02)	14
Ellipsoidal	5.31E+00 (1.48E+01)	6	1.98E+01 (6.54E+00)	0	1.57E-07 (1.92E-07)	30	9.92E+00 (6.00E+00)	0
Dejong's	1.28E+01 (8.46E+00)	0	1.78E+01 (7.79E+00)	0	1.41E+01 (7.00E+00)	0	1.73E+01 (9.39E+00)	0
Rosenbrock	1.84E-01 (3.83E-01)	20	2.46E+00 (7.19E+00)	4	4.32E+01 (2.95E+01)	2	1.23E+03 (5.67E+02)	0
step	6.24E-09 (9.58E-09)	30	5.06E-02 (4.54E-02)	16	6.74E-32 (2.56E-31)	30	1.20E+01 (6.67E+00)	0
Rastrigin	1.85E+00 (4.00E+00)	24	2.99E-02 (2.87E-02)	22	8.43E+00 (2.22E+00)	0	2.08E+01 (5.67E+00)	0
Neumair	1.80E+01 (3.12E+00)	30	6.41E+01 (6.45E+01)	28	1.43E+03 (1.47E+03)	6	1.33E+04 (6.59E+03)	0

6.1 First way of analysis

6.1.1 EPX-MPTM vs LX-MPTM, SBX-MPTM, HX-MPTM

The average value of the objective function, standard error and the number of successful runs are listed in Table 3. Results indicate that the proposed operator, EPX-MPTM, has a smaller value of the mean objective function in all problems. The proposed operator has 100 % success rate in twelve problems while the other operators solve only ten problems. None of the GAs could solve P14 in all runs.

6.1.2 EPX-NUM vs LX-NUM, SBX-NUM, HX-NUM

The proposed operator EPX-NUM is also compared with other crossover operators in conjunction with NUM mutation operator. In 15 problems, the EPX-NUM shows more near

Table 7 Mean, standard deviation and number of succesful runs of LogX and EPX for 1000 simulated results with 30 independent runs

Problem name	LogX-PM			EPX-PM		
	Mean	SD	Successful runs	Mean	SD	Successful runs
Cigar	8.56E+04	7.07E+04	0	6.05E+04	1.83E+05	0
New	5.37E+00	8.09E+00	6	2.40E-03	4.30E-03	30
Greiwank	1.05E+00	9.40E-02	0	3.15E-01	3.41E-01	0
Cosine Mixture	-2.98E+00	4.32E-02	30	-2.99E+00	6.00E-03	30
Brown	1.40E-03	1.20E-03	30	7.88E-04	1.60E-03	30
Generalized1	9.10E-03	2.78E-02	28	1.39E-05	1.92E-05	30
Generalized2	2.64E-02	5.77E-02	30	3.30E-03	4.30E-03	30
Sphere	2.96E-02	4.34E-02	24	7.87E-04	1.20E-03	30
New 13	2.23E-02	1.75E-02	28	4.92E-01	5.24E-01	10
Hyper Ellipsoid	3.72E-01	4.07E-01	4	1.06E-01	2.37E-01	22
LevyMount1	8.60E-03	2.73E-02	28	2.89E-05	4.89E-05	30
LevyMount2	1.73E-02	1.74E-02	28	1.90E-03	4.00E-03	30
Ellipsoidal	1.42E+01	5.52E+00	0	5.18E-01	9.47E-01	8
Dejong's	9.28E+00	7.70E+00	0	1.25E+01	8.82E+00	0
Rosenbrok	1.06E+02	9.53E+01	0	3.30E+02	3.55E+02	0
step	9.02E-126	3.49E-125	30	1.70E-13	6.60E-13	30
Rastrigin	2.07E+01	1.28E+01	0	7.21E+00	3.20E+00	0
Neumair	5.44E+03	5.57E+03	0	2.44E+03	2.42E+03	0

optimal results than other crossover operator i.e., LX-NUM, SBX-NUM and HX-NUM. In other three problems, there is not much difference in the results (Table 4).

6.1.3 EPX-PM vs LX-PM, SBX-PM, HX-PM

Table 5 shows the mean, standard deviation and the number of successful runs for third group of comparison. The proposed operator outperforms the other crossover operators in all problems except P4. EPX produces more optimal results than the other crossover operators. From the above comparison, it can be concluded that on the basis of mean, standard deviation (S.D) and successful rate, EPX worked efficiently with MPTM, NUM and PM operators. It resulted in more near optimal results than other crossover operators in most of the test problems.

6.1.4 EPX-MPTM vs LogX-MPTM, EPX-NUM vs LogX-NUM and EPX-PM vs LogX-PM

The proposed EPX is compared with LogX operator in conjunction with MPTM, NUM and PM as mutation operators (Tables 6 and 7). Results show the better performance of EPX in terms of less average and standard deviation values. The EPX solves all problems successfully than the LogX operator as indicated by the successful runs mentioned in tables 6 and 7.

6.2 Second way of analysis

The second way of analysis is to compare the relative performance of all the GAs simultaneously. The performance index (PI) is calculated in the following manner:

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(t_1\alpha_1^i + t_2\alpha_2^i + t_3\alpha_3^i) \tag{26}$$

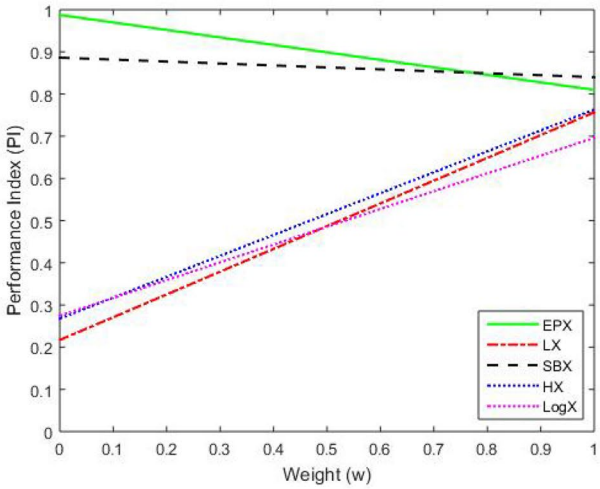
where

$$\alpha_1^i = \frac{Sr^i}{Tr^i} \tag{27}$$

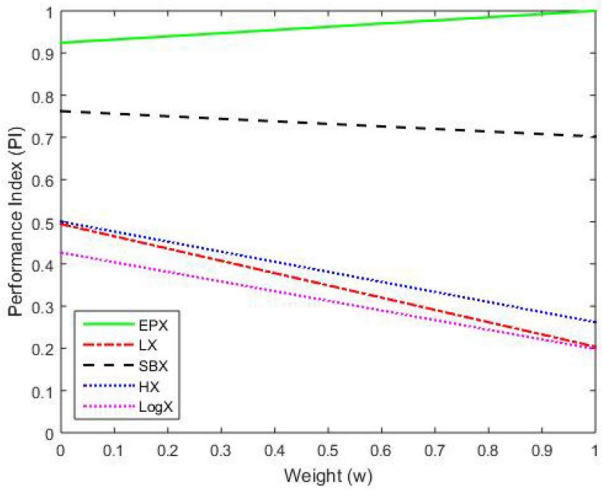
$$\alpha_2^i = \frac{Mf^i}{Lmf^i} \tag{28}$$

$$\alpha_3^i = \frac{Sf^i}{Lsf^i} \tag{29}$$

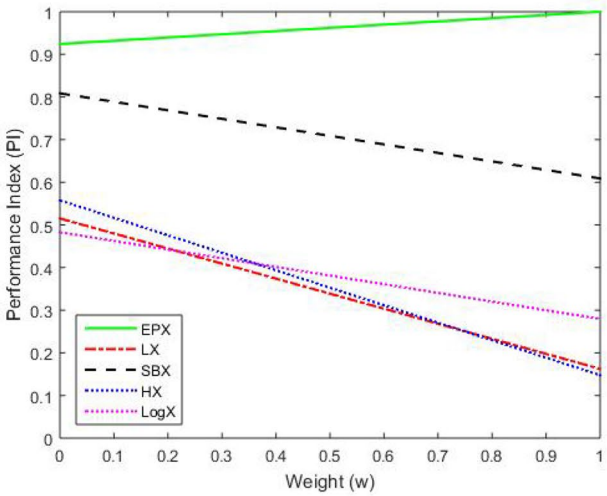
where $i = 1, 2, ..., N_p$; Sr^i is the number of successful runs of i^{th} problem; Tr^i is the total number of runs of i^{th} problem; Mf^i is the mean objective function value of i^{th} problem; Lmf^i is the least mean objective function value of i^{th} problem; Sf^i is the standard deviation of i^{th} optimization problem; Lsf^i is the least standard deviation value among all GAs of i^{th} optimization problem and N_p is the total number of problems analyzed.



(a) PI when t_1=w and $t_2 = t_3$=(1-w)/2



(b) PI when t_2=w and $t_1 = t_3$=(1-w)/2



(c) PI when t_3=w and $t_1 = t_2$=(1-w)/2

Fig. 2 **a** PI when $t1$=w and $t2 = t3$=(1-w)/2, **b** PI when $t2$=w and $t1 = t3$=(1-w)/2, **c** PI when $t3$=w and $t1 = t2$=(1-w)/2.

t_1, t_2 and t_3 are the weights assigned to successful runs, mean objective function value and the standard deviation of the objective function value respectively. Same weights are assigned to two terms at a time so that the behavior of PI can be easily analyzed. The following three cases are considered here:

Case 1: $t_1 = w, t_2 = t_3 = \frac{1-w}{2}, \quad 0 \le w \le 1$

Case 2: $t_2 = w, t_1 = t_3 = \frac{1-w}{2}, \quad 0 \le w \le 1$

Case 3: $t_3 = w, t_1 = t_2 = \frac{1-w}{2}, \quad 0 \le w \le 1$

From Fig. 2, it is evident that the proposed crossover operator i.e., EPX outperforms the other crossover operators.

6.3 Third way of analysis

The student's t test has also been applied to test the hypothesis that whether the mean of proposed operator is smaller than the other three GAs. It further investigate whether the mean of the proposed algorithm is more close to the optimal value than the other crossover operators. Results are shown in Table 8.

The GA convergence performed on P9 (Axis parallel hyper ellipsoid) is shown in Figs. 3, 4, 5. LX and HX take more generations to converge i.e. have more population diversity and SBX converges too quickly because of more selection pressure. EPX works more effectively than LogX operator as it attains the global optima while maintaining the balance between exploration and exploitation. It can be

Table 8 t-Statistic result

t-Statistic result			
Function name	EPX vs LX	EPX vs SBX	EPX vs HX
Cigar	1	1	1
New	1	0	1
Greiwank	1	1	1
Cosine mixture	0	0	1
Brown	1	1	1
Generalized1	1	0	1
Generalized2	1	1	1
Sphere	1	1	1
New 13	1	1	1
Hyper ellipsoid	1	0	1
LevyMount1	1	1	1
LevyMount2	1	1	1
Ellipsoidal	1	0	1
Dejong's	0	0	1
Rosenbrock	1	1	1
step	1	1	0
Rastrigin	1	1	1
Neumair	1	1	1

analyzed on any test problem for all competing selection strategies (Table 8).

7 Conclusion

In this paper, we proposed a new real coded GA, the "Exponentiated Pareto" crossover. It is used in conjunction with MPTM, NUM and PM mutation operators. Graphical representation shows that the proposed operator generates near parent offspring for a smaller value of the spread parameter. The problem dimensions are fixed to be 30 as used in earlier studies.

For the sake of analysis, the GAs are categorized into three groups. In the first group, the EPX operator is compared with other crossover operators in conjunction with MPTM mutation operator, the second group compares the varying crossover operators with NUM mutation operator and the third group use the PM mutation operator with the four varying crossover operators.

Three kind of analysis have been performed. In the first way of analysis, the optimal fitness values of the EPX operator are compared with other crossover operators. The mean and standard deviation results show that the EPX operator is more efficient then the other crossover operators. It has 100% success rate in almost all problems.

The second kind of analysis is carried out by using the performance index. Figure 2 shows that the EPX operator

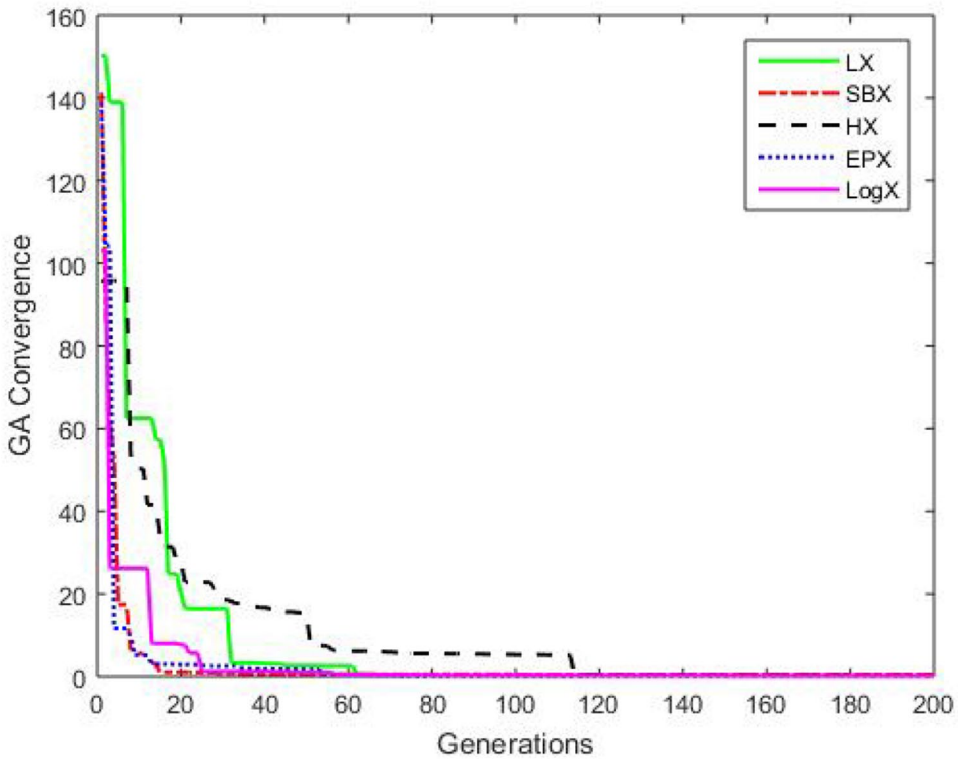


Fig. 3 GA Convergence with EPX-MPTM and other crossover operators for P9

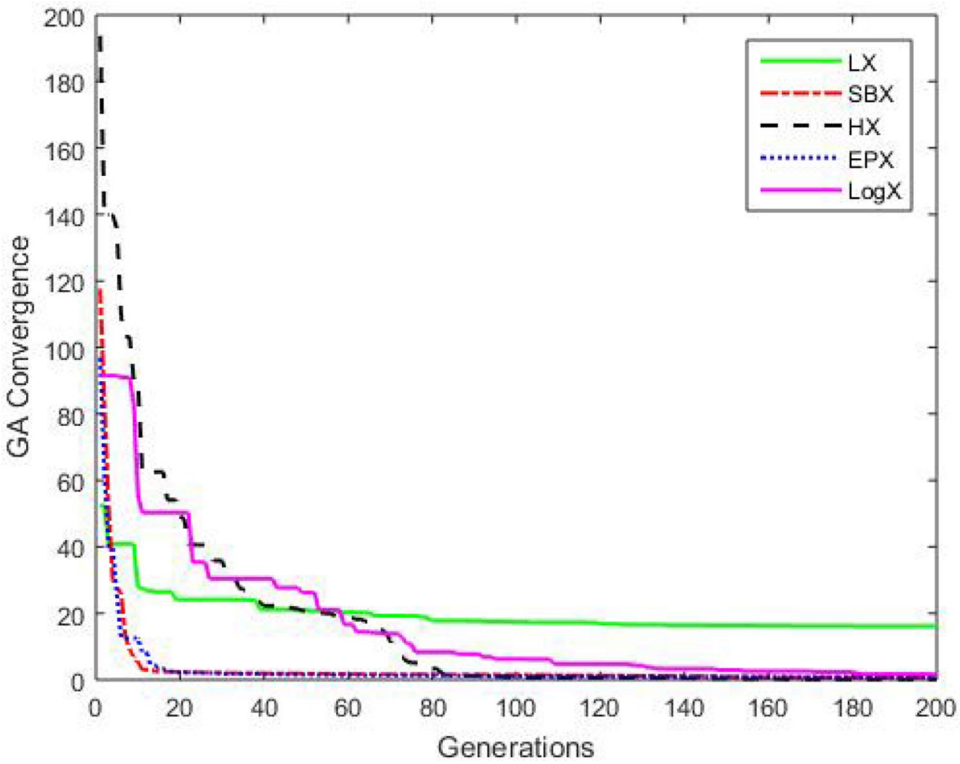


Fig. 4 GA Convergence with EPX-NUM and other crossover operators for P9

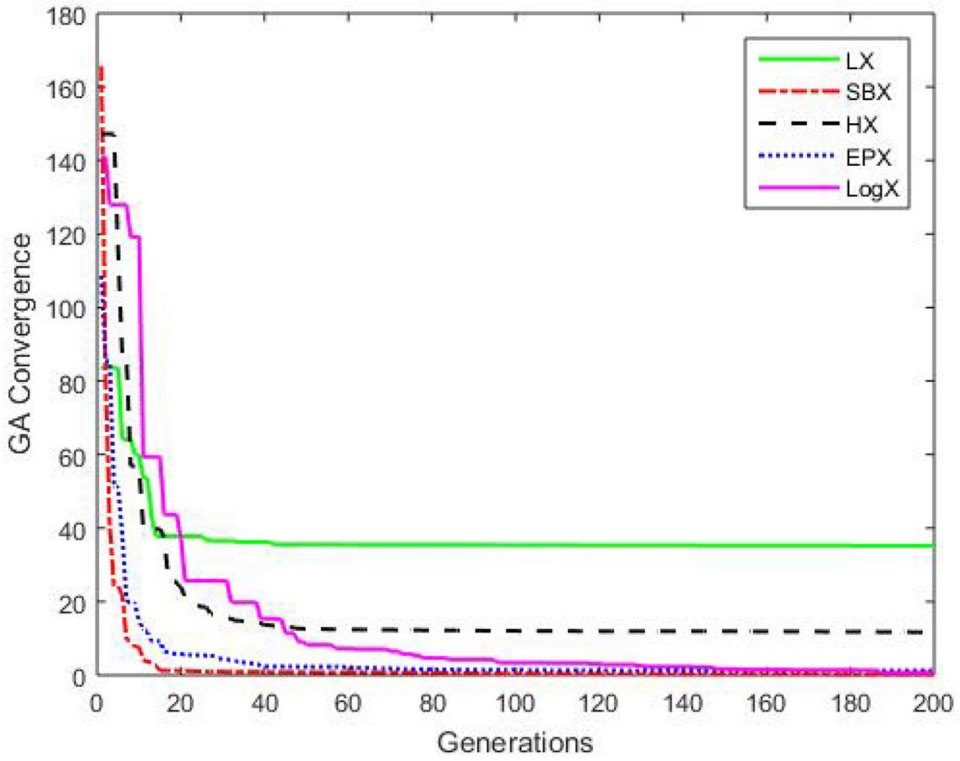


Fig. 5 GA Convergence with EPX-PM and crossover operators for P9

outperforms the other crossover operators. Finally, the significant t-test results support our hypothesis that the EPX provides more near optimal results.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12065-021-00636-4.

References

1. Holland J (1975) Adaptation in natural and artificial systems: an introductory analysis with application to biology. Control and Artificial Intelligence. MIT press, Cambridge
2. Zheng SR, Lai JM, Liu GL, Gang T (2006) Improved real coded hybrid genetic algorithm. Comput Appl 26(8):1959–1962
3. Liu HH, Cui C, Chen J (2013) An improved genetic algorithm for solving travel salesman problem. Trans Beijing Inst Technol 33(4):390–393
4. Jingi W, Yang X, Lei C (2012) The application of GA-based PID parameter optimization for the control of superheated steam temperature. In: International conference on machine learning and cybernetics. vol 3, pp 835–839
5. Golberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Publishing Company, Boston
6. Deb K (2001) Nonlinear goal programming using multi-objective genetic algorithms. J Op Res Soc 52(3):291–302
7. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Sci 220(4598):671–680
8. Price K, Storn RM, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer Science and Business Media, Berlin
9. Eberhart RC, Shi Y, Kennedy J (2001) Swarm intelligence. Elsevier, Netherlands
10. Bäck T, Schwefel HP (1993) An overview of evolutionary algorithms for parameter optimization. Evol Comput 1(1):1–23
11. Chen CT, Wu CK, Hwang C (2008) Optimal design and control of CPU heat sink processes. IEEE Trans Compon Packag Technol 31(1):184–195
12. Chen CT, Chuang YC (2010) An intelligent run-to-run control strategy for chemical-mechanical polishing processes. IEEE Trans Semicond Manuf 23(1):109–120
13. Dyer JD, Hartfield RJ, Dozier GV, Burkhalter JE (2012) Aerospace design optimization using a steady state real-coded genetic algorithm. Appl Math Comput 218(9):4710–4730
14. Tsai CW, Lin CL, Huang CH (2010) Microbrushless DC motor control design based on real-coded structural genetic algorithm. IEEE/ASME Trans Mech 16(1):151–159
15. Valarmathi K, Devaraj D, Radhakrishnan TK (2009) Real-coded genetic algorithm for system identification and controller tuning. Appl Math Model 33(8):3392–3401
16. Goldberg DE (1990) Real-coded genetic algorithms, virtual alphabets and blocking. University of Illinois at Urbana Champaign, Champaign
17. Lawrence D (1991) Handbook of genetic algorithms. Van Nostrand Reinhold
18. Wright AH (1991) Genetic algorithms for real parameter optimization. Found Genet Algorithms 1:205–218
19. Janikow CZ, Michalewicz Z (1991) An experimental comparison of binary and floating point representations in genetic algorithms. ICGA
20. Hussain A, Muhammad YS (2020) Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. Complex Intell Sys 6(1):1–14
21. Eiben AE, Schut MC, de Wilde AR (2006) Is self-adaptation of selection pressure and population size possible?–A case study. In: Parallel problem solving from nature-PPSN IX, pp 900–909
22. Michalewicz Z, Logan T, Swaminathan S (1994) Evolutionary operators for continuous convex parameter spaces. In Proceedings of the 3rd annual conference on evolutionary programming, PP 84–97
23. Radcliffe NJ (1991) Equivalence class analysis of genetic algorithms. Complex Syst 5(2):183–205
24. Mühlenbein H, Schlierkamp-Voosen D (1993) Predictive models for the breeder genetic algorithm in continuous parameter optimization. Evol Comput 1(1):25–49
25. Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. Found Genet Algorithms 2:187–202
26. Michalewicz Z, Janikow CZ (1991) Handling constraints in genetic algorithms. ICGA 151–157
27. Voigt HM (1992) Fuzzy evolutionary algorithms. International Computer Science Institute
28. Voigt HM, Mühlenbein H, Cvetkovic D (1995) Fuzzy recombination for the breeder genetic algorithm. In: Proceedings of sixth international conferene on genetic algorithms
29. Tsutsui S, Yamamura M, Higuchi T (1999) Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: Proceedings of the 1st annual conference on genetic and evolutionary computation, vol 1, pp 657–664
30. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Syst 9(2):115–148
31. Tutkun N (2009) Optimization of multimodal continuous functions using a new crossover for the real-coded genetic algorithms. Expert Syst Appl 36(4):8172–8177
32. Deep K, Thakur M (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188(1):895–911
33. Ono I, Kita H, Kobayashi S (2003) A real-coded genetic algorithm using the unimodal normal distribution crossover. In: Advances in evolutionary computing, pp 213–237
34. Ono I, Kita H, Kobayashi S (1999) A robust real-coded genetic algorithm using unimodal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In: Proceedings of the 1st annual conference on genetic and evolutionary computation. vol 1, pp 496–503
35. Deb K, Anand A, Joshi D (2002) A computationally efficient evolutionary algorithm for real-parameter optimization. Evol Comput 10(4):371–395
36. Sinha A, Tiwari S, Deb K (2005) A population-based, steady-state procedure for real-parameter optimization. IEEE Congr Evol Comput 1:514–521
37. Ling SH, Leung FH (2007) An improved genetic algorithm with average-bound crossover and wavelet mutation operations. Soft Comput 11(1):7–31
38. Herrera F, Lozano M, Sanchez AM (2003) A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. Int J Intell Syst 18(3):309–338
39. Mäkinen RA, Périaux J, Toivanen J (1999) Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. Int J Numer Methods Fluids 30(2):149–159
40. Miettinen K, Mäkelä MM, Toivanen J (2003) Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. J Global Optim 27(4):427–446
41. Michalewicz Z (2013) Genetic algorithms+ data structures= evolution programs. Springer Science and Business Media, Berlin
42. Michalewicz Z (1995) Genetic algorithms, numerical optimization, and constraints. In: Proceedings of the sixth international conference on genetic algorithms. vol 195, pp 151–158
43. Ali MZ, Awad NH, Suganthan PN, Shatnawi AM, Reynolds RG (2018) An improved class of real-coded Genetic Algorithms for numerical optimization. Neurocomput 275:155–166

44. Jin YF, Yin ZY, Shen SL, Zhang DM (2017) A new hybrid real-coded genetic algorithm and its application to parameters identification of soils. Inverse Probl Sci Eng 25(9):1343–1366
45. Pattanaik JK, Basu M, Dash DP (2018) Improved real coded genetic algorithm for dynamic economic dispatch. J Electr Syst Inf Technol 5(3):349–362
46. Elsayed SM, Sarker RA, Essam DL (2014) A new genetic algorithm for solving optimization problems. Eng Appl Artif Intell 27:57–69
47. Al-Naqi A, Erdogan AT, Arslan T (2013) Adaptive three-dimensional cellular genetic algorithm for balancing exploration and exploitation processes. Soft Comput 17(7):1145–1157
48. Ahmad I, Almanjahie IM (2020) A novel parent centric crossover with the log-logistic probabilistic approach using multimodal test problems for real-coded genetic algorithms. Math Probl Eng 2020. https://doi.org/10.1155/2020/2874528
49. Wang J, Cheng Z, Ersoy OK, Zhang P, Dai W, Dong Z (2018) Improvement analysis and application of real-coded genetic algorithm for solving constrained optimization problems. Math Probl Eng 2018. https://doi.org/10.1155/2018/5760841
50. Chuang YC, Chen CT, Hwang C (2015) A real-coded genetic algorithm with a direction-based crossover operator. Inf Sci 305:320–348
51. Das AK, Pratihar DK (2019) A directional crossover (DX) operator for real parameter optimization using genetic algorithm. Appl Intell 49(5):1841–1865
52. Das AK, Pratihar DK (2020) A direction-based exponential crossover operator for real-coded genetic algorithm. In: Singh B, Roy A, Maiti D (eds) Recent advances in theoretical, applied, computational and experimental mechanics. Lecture Notes in Mechanical Engineering. Springer, Singapore
53. Chuang YC, Chen CT, Hwang C (2016) A simple and efficient real-coded genetic algorithm for constrained optimization. Appl Soft Comput 38:87–105
54. Zhao Y, Cai Y, Cheng D (2017) A novel local exploitation scheme for conditionally breeding real-coded genetic algorithm. Multimed Tools Appl 76(17):17955–17969
55. Rolland L, Chandra R (2016) The forward kinematics of the 6–6 parallel manipulator using an evolutionary algorithm based on generalized generation gap with parent-centric crossover. Robotica 34(1):1
56. Da Ronco CC, Benini E (2013) A simplex crossover based evolutionary algorithm including the genetic diversity as objective. Appl Soft Comput 13(4):2104–2123
57. Naqvi FB, Yousaf Shad M, Khan S (2021) A new logistic distribution based crossover operator for real-coded genetic algorithm. J Stat Comput Simul 91(4):817–835