



A directional crossover (DX) operator for real parameter optimization using genetic algorithm

Amit Kumar Das¹ · Dilip Kumar Pratihar¹ 

Published online: 13 December 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Nature-inspired optimization algorithms have received more and more attention from the researchers due to their several advantages. Genetic algorithm (GA) is one of such bio-inspired optimization techniques, which has mainly three operators, namely selection, crossover, and mutation. Several attempts had been made to make these operators of a GA more efficient in terms of performance and convergence rates. In this paper, a directional crossover (DX) has been proposed for a real-coded genetic algorithm (RGA). As the name suggests, this proposed DX uses the directional information of the search process for creating the children solutions. Moreover, one method has been suggested to obtain this directional information for identifying the most promising areas of the variable space. To measure the performance of an RGA with the proposed crossover operator (DX), experiments are carried out on a set of six popular optimization functions, and the obtained results have been compared to that yielded by the RGAs with other well-known crossover operators. RGA with the proposed DX operator (RGA-DX) has outperformed the other ones, and the same has been confirmed through the statistical analyses. In addition, the performance of the RGA-DX has been compared to that of other five recently proposed optimization techniques on the six test functions and one constrained optimization problem. In these performance comparisons also, the RGA-DX has outperformed the other ones. Therefore, in all the experiments, RGA-DX has been found to yield the better quality of solutions with the faster convergence rate.

Keywords Evolutionary algorithm · Real-coded genetic algorithm · Directional crossover · Exponential crossover · Convergence rate

1 Introduction

An optimization is the process of searching the best feasible solution among several possibilities. Most of the real-world optimization problems are generally non-linear in nature and they have one or more number of design variables and constraints. The need for optimizing the design of a product or a process is to reduce the total cost, material consumption, waste and time, or to make an improvement in performance, profit and benefit [1–6]. The traditional deterministic approaches do not involve any randomness in the algorithm and

consequently, they behave in a mechanical deterministic way. These techniques are not suitable to solve complex optimization problems, especially when the objective functions are found to be non-smoothed or ill-conditioned. Therefore, researchers had looked to the Mother Nature to get innovative ideas for problem-solving. Since then, various stochastic approaches, inspired by several natural phenomena, have been designed and successfully applied to solve a variety of problems in different spheres of engineering and industrial applications [7–18]. These techniques utilize random numbers and due to which, different optimized results can be obtained in several runs with the same initial population. Furthermore, there are several advantages of stochastic approaches over the deterministic ones, such as flexibility in defining objective and constraint functions, ability to avoid local optima problem, can be implemented for solving any kind of optimization problems, and others. Due to several benefits of the stochastic methods, researchers have made their efforts to contribute not only to the application related problems but also to the theoretical studies of the same.

✉ Dilip Kumar Pratihar
dkpra@mech.iitkgp.ac.in
Amit Kumar Das
amit.besus@gmail.com

¹ Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

In literature, the theoretical studies on the stochastic methods have been broadly grouped into three major categories. The first one deals with the improvement in performances of the current algorithms using the modified or newly developed stochastic operators [19–23], while several optimization techniques are hybridized to obtain the better performances [24–27] compared to the individual ones in the second category. In the third one, researchers have proposed several new metaheuristic techniques inspired by various natural phenomena and others. Cricket Algorithm (CA) [28], Grasshopper Optimization algorithm (GOA) [29], Sine Cosine Algorithm (SCA) [30], Social Spider Optimization (SSO) [31], Search Group Algorithm (SGA) [32], etc. are a few worth-mentioning examples of the newly-proposed optimization techniques.

Among several stochastic techniques, Genetic algorithm (GA) is one of the most widely used optimization techniques and it has been implemented to solve a variety of problems in various domains [33–36]. GA was proposed by Holland [37] and it imitates the basic principle of natural selection, which is nothing but the survival of the fittest. It is a population-based global optimization technique and it has several advantages, such as it is easy and straight-forward to apply, capable of tackling a large search space and there is no need of the detailed mathematical derivations for defining the objective and functional constraints of the problem. There are several coding schemes available for a GA (binary-coded GA [38], gray-coded GA [39], integer genes [40], and real-coded GA [20, 41]) in the literature. However, the real-coded genetic algorithm (RGA) has been found to be more popular [42] compared to the other ones for its inherent capability to provide the solutions in the numeric form directly and there is no need to code and decode the solutions. Moreover, many real-world applications of RGA (especially, for high accuracy problems with very large search spaces) [43, 44] have demonstrated the superiority of using the same compared to that of the other coding schemes.

A standard GA mainly consists of three stochastic operators, such as selection, crossover, and mutation. Each of these has its own operating mechanism to search for the optimal solution. In the selection scheme, more potential candidates in terms of their fitness values are chosen in the mating pool, while during the crossover, the properties of the parents are transformed into the newly created offspring. After the crossover, mutation is applied to bring a sudden change in the properties of the created offspring and this helps a GA to avoid the local minimum trapping problem. A standard GA operates in a cyclic manner with these three mentioned operators, until it satisfies the stopping criterion. In the literature of GA, several contributions have been made to improve the performance of this optimization technique and among them, a greater focus has been seen in proposing new crossover schemes or improving the existing ones. This said fact

signifies that the crossover is the most important recombination operator [45], which greatly helps a GA to reach the globally optimum point. In the present study, a new crossover operator has been proposed to improve the overall performances of a real-coded genetic algorithm (RGA).

2 Literature review

To make the GA more efficient, several new crossover schemes had been developed by various researchers. Wright [46] introduced a heuristic mating approach, where an offspring is yielded from two mating parents with a biasness towards the better one. The concept of flat crossover was proposed by Radcliffe [47], where the offspring is created randomly between a pair of solutions. The arithmetical crossover was proposed by Michalewicz [48] with its two versions, namely uniform and non-uniform arithmetical crossover. The idea of a blend crossover (BLX- α) was proposed by Eshelman and Schaffer [49]. In this case, the positions of the offspring are determined by the value of the parameter α and the recommended value for the parameter was found to be equal to 0.5. A fuzzy recombination operator was introduced by Voigt et al. [50], where the algorithm converged linearly for a large class of unimodal functions.

The concept of the Simulated Binary Crossover (SBX) was proposed by Deb and Agrawal [51]. This mimics the idea of a single-point crossover operation on binary chromosomes in continuous search space. It creates two children solutions out of two mating parents. A unimodal normally distributed crossover (UNDX) operator was designed by Ono and Kobayashi [52], where two or more offsprings were supposed to be generated using three mating solutions. A steady-state GA with the said operator was implemented on three difficult-to-solve problems. Furthermore, an effort was made to improve the performance of the UNDX by Ono et al. [53] with the aid of a uniform crossover scheme. However, only three problems were used to demonstrate the performance of the improved UNDX operator. Later, this work was further extended and presented as a multi-parental UNDX [54], where more than three parent solutions were supposed to participate in a crossover. However, there was no significant improvement reported in the performance of a GA with the extended version of the UNDX compared to the original one.

Herrera and Lozano [55] developed a real-coded crossover mechanism with the help of a fuzzy logic controller to avoid the premature convergence issue of a GA. The extension of this work was also found in [56], where the performance of an RGA with the proposed operator was found to be reasonable. However, the number of problems in the experiment was not sufficient to conclude anything deterministically. The theory of a simplex crossover (SPX) was given by Tsutui et al. [57], in which the children solution vector was generated by the

uniformly sampling values deduced from simplex formed by the parent vectors. It was reported that this method works fine on the multimodal problems having more than one local optimum basins.

Apart from these, Deb et al. [58] introduced a parent-centric crossover (PCX) and its performance was tested on a set of three optimization problems. The results were compared to that of the other two crossover operators, such as SPX and UNDX and a few metaheuristic algorithms. The Laplace crossover (LX), proposed by Deep and Thakur [59], uses the Laplace distribution to determine the positions of the newly created offspring. In that paper, the performance of the LX was compared only to that of heuristic crossover (HX). Later, a directed crossover was proposed by Kuo and Lin [60] and the mechanism of this operator was developed utilizing the reflection and expansion searching approach of Nelder-Mead's simplex method. Lim et al. [61] proposed the idea of a Rayleigh crossover (RX) based on the Rayleigh distribution. The performance of an RGA with this crossover operator was found to be better, when it was compared to that of an RGA with LX operator. However, this comparison was not sufficient enough to establish the superiority of this operator. Recently, Chuang et al. [45] introduced a parallel-structured RGA. They called it RGA-RDD and this is an ensemble of three evolutionary operators, such as Ranking Selection (RS), Direction-Based Crossover (DBX), and Dynamic Random Mutation (DRM). From the various results yielded by the RGA-RDD, it was concluded that this was more suitable in solving high-dimensional and multi-modal optimization problems. Das and Pratihar [41] developed a direction-based exponential crossover operator, which was influenced by the directional information of the problem and it used exponential functions to create offspring.

From the above literature survey of the GA, it has been observed that there exist a number of crossover schemes with their inherent advantages and disadvantages. However, it is to be noted that in most of the crossover operators (except DBX in RGA-RDD), the concept of obtaining the most promising search direction and utilizing this information for creating the children solutions is missing. In DBX, one procedure had been developed to use the directional information. However, there is no concept of using directional probability (p_d) term, which helps the GA to maintain a proper balance between the selection pressure and population diversity. These concepts are introduced in a more efficient way in this study and these are the novelties of this proposed DX operator. Furthermore, it is all accredited that there is no end of improvements. In addition, considering the insights of the theorem "No Free Lunch" [62], it can be stated that it is impossible for an algorithm to provide the best results for all types of problems. In another way, it can be said that a superior performance of an algorithm on a specific set of problems does not guarantee an equal performance in solving all other problems. This fact

motivates the researchers to contribute by either improving the existing algorithms or proposing new kind of algorithms. Moreover, this fact is also the source of motivation for the present study, where a new directional crossover (DX) has been proposed for a real-coded genetic algorithm. It is to be noted that the present work is quite different from the previous work of the authors [41] related to crossover operation. The rest of the text has been arranged as follows: Section 3 provides the details of the proposed crossover operator, while the results and discussion are included in Section 4. A constrained engineering problem has been solved using an RGA with the proposed DX operator in Section 5 and finally, some concluding remarks are made in Section 6.

3 The proposed directional crossover operator

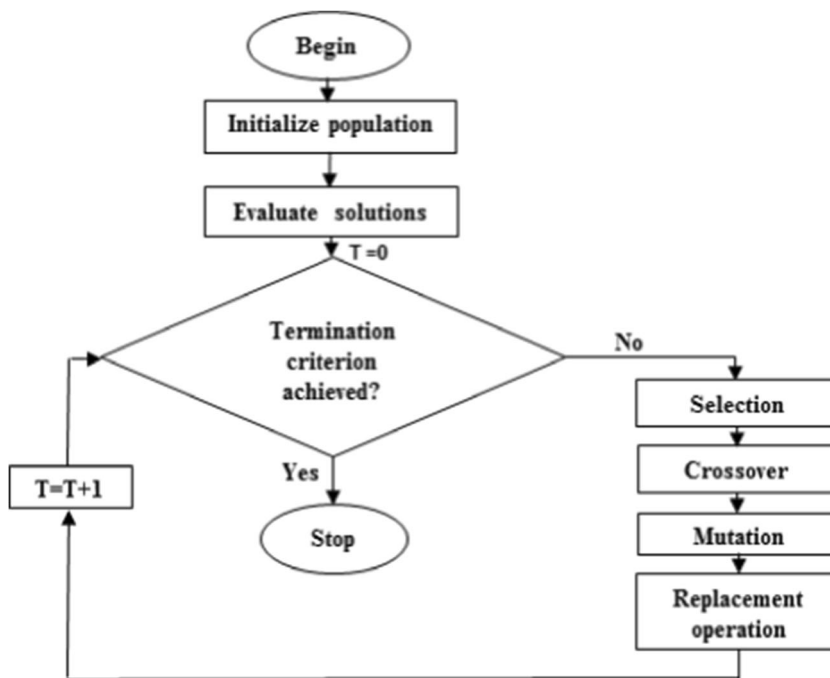
Here, we present a new directional crossover (DX) operator for the real-coded genetic algorithm (RGA). Figure 1 shows the flowchart of an RGA. As the name suggests, the proposed crossover scheme is guided by the directional information of the optimization problem during an evolution. The directional information is nothing but the prior knowledge about the most-promising areas in the search space, where the probability of finding a better solution is more. In this proposed crossover operator, two new children solutions are created after exchanging the properties of two mating parents. Furthermore, the directional information is obtained for a crossover operation by comparing the mean position of the two mating parents with the position of the current best solution. The details of this proposed crossover operator are described in Section 3.1.

3.1 Detailed descriptions of DX

Let us consider, the population size of an optimization algorithm is N and the number of variables of the problem is d . After the selection process is over, $\frac{N}{2}$ number of mating pairs are formed randomly from the N selected solutions. Now, a particular pair of solutions is allowed to participate in crossover, if a random number, created between 0 and 1, is found to be either less than or equal to the crossover probability (p_c). Otherwise, there will be no modification in those two mating parents.

Now, assume two mating parents, say p_1 and p_2 , are allowed to participate in the crossover. As the proposed operator is implemented variable-wise, another probability, namely variable-wise crossover probability (p_{cv}), is used to determine the occurrence of the crossover for a certain variable position. Moreover, the concept of the parameter p_{cv} is used in the algorithm for a variable in the same way as in the case of

Fig. 1 Flowchart of a standard RGA



p_c . This means that the i^{th} -variable of the mating solutions p_1 and p_2 (i.e., p_1^i and p_2^i , respectively), are going to participate in crossover, only when a random number, ranging from 0 to 1, is seen to be either less than or equal to that of p_{cv} . Otherwise, no changes are made in the i^{th} -variable of the mating parents. Here, i varies from 1 to d .

Next, if the two mating solutions (p_1^i and p_2^i) are allowed to participate in the crossover, there can be two distinct situations and depending upon the situations, the crossover mechanisms are going to be different. The first situation occurs, when the pair of solutions (p_1^i and p_2^i) is found to be unequal, while in the second situation, the said solutions are seen to have the same values. The crossover mechanisms of both the situations are explained as follows:

First situation: In this case, the two mating parents (p_1^i and p_2^i) are seen to have different values. The mean of the two mating parents (say, p_{mean}^i) is calculated. Considering p_{best} as the current best solution of the population in terms of the fitness value, p_{mean}^i is compared to that of the i^{th} -variable of the p_{best} (i.e., p_{best}^i) to obtain the directional information. It is assumed that in the direction of the p_{best}^i , the chances of creating good solutions from the mating parents are more. Now, the locations of the newly created children solutions are influenced by this prior information of direction with a probability value, namely directional probability (p_d). Due to fact that there is no guarantee to get the better offspring in the obtained direction, the parameter p_d is introduced in the proposed operator. Now, two intermediate parameters, such as val

and β , are calculated using Eqs. (1) and (2). Moreover, when the value of the p_{best}^i is found to be either greater than or equal to p_{mean}^i , the children solutions (say, c_1 and c_2) are created as follows:

$$val = 1 - (0.5)^{e^{\left[\frac{|p_1^i - p_2^i|}{(V_u^i - V_l^i)} \right]}} \tag{1}$$

$$\beta = \frac{r}{\alpha^2} \tag{2}$$

$$\left. \begin{aligned} c_1 &= val \times (p_1^i + p_2^i) + \alpha^r \times e^{(1-\beta)} \times (1-val) \times |p_1^i - p_2^i| \\ c_2 &= (1-val) \times (p_1^i + p_2^i) - \alpha^{(1-r)} \times e^{(-\beta)} \times val \times |p_1^i - p_2^i| \end{aligned} \right\} \text{ if } r_1 \leq p_d, \tag{3, 4}$$

$$\left. \begin{aligned} c_1 &= val \times (p_1^i + p_2^i) - \alpha^r \times e^{(1-\beta)} \times (1-val) \times |p_1^i - p_2^i| \\ c_2 &= (1-val) \times (p_1^i + p_2^i) + \alpha^{(1-r)} \times e^{(-\beta)} \times val \times |p_1^i - p_2^i| \end{aligned} \right\} \text{ if } r_1 > p_d, \tag{5, 6}$$

where r and r_1 are the two different random numbers generated in the range of 0 and 1. V_u^i and V_l^i are the upper and lower limits of the i^{th} -variable, respectively. α is the multiplying factor and it is a user-defined parameter. In another case, where p_{best}^i is seen to be less than p_{mean}^i , the offsprings are generated as follows:

$$\left. \begin{aligned} c_1 &= val \times (p_1^i + p_2^i) - \alpha^r \times e^{(1-\beta)} \times (1-val) \times |p_1^i - p_2^i| \\ c_2 &= (1-val) \times (p_1^i + p_2^i) + \alpha^{(1-r)} \times e^{(-\beta)} \times val \times |p_1^i - p_2^i| \end{aligned} \right\} \text{ if } r_1 \leq p_d, \tag{7, 8}$$

$$\begin{aligned} c_1 &= val \times (p_1^i + p_2^i) + \alpha^r \times e^{(1-\beta)} \times (1-val) \times |p_1^i - p_2^i| \\ c_2 &= (1-val) \times (p_1^i + p_2^i) - \alpha^{(1-r)} \times e^{(-\beta)} \times val \times |p_1^i - p_2^i| \end{aligned} \quad \text{if } r_1 > p_d, \tag{9, 10}$$

Second situation: In this case, the values of p_1^i and p_2^i are found to be the same. Therefore, the value of p_{mean}^i is going to be equal to that of either p_1^i or p_2^i . In certain cases, where p_{mean}^i is found to have the equal value compared to p_{best}^i , no crossover takes place between these two mating solutions. Otherwise, new children solutions are yielded using the following equations:

$$val = 1 - (0.5)^e \left[\frac{|p_{best}^i - p_{mean}^i|}{(V_u^i - V_l^i)} \right], \tag{11}$$

$$\beta = \frac{r}{\alpha^2}, \tag{12}$$

$$\begin{aligned} c_1 &= val \times (p_{best}^i + p_{mean}^i) + \alpha^r \times e^{(1-\beta)} \times (1-val) \times (p_{best}^i - p_{mean}^i) \\ c_2 &= (1-val) \times (p_{best}^i + p_{mean}^i) - \alpha^{(1-r)} \times e^{(-\beta)} \times val \times (p_{best}^i - p_{mean}^i) \end{aligned} \quad \text{if } r_1 \leq p_d, \tag{13, 14}$$

$$\begin{aligned} c_1 &= val \times (p_{best}^i + p_{mean}^i) - \alpha^r \times e^{(1-\beta)} \times (1-val) \times (p_{best}^i - p_{mean}^i) \\ c_2 &= (1-val) \times (p_{best}^i + p_{mean}^i) + \alpha^{(1-r)} \times e^{(-\beta)} \times val \times (p_{best}^i - p_{mean}^i) \end{aligned} \quad \text{if } r_1 > p_d, \tag{15, 16}$$

where r and r_1 are again two different random numbers created in between 0 and 1. Moreover, similar to the first situation, val and β are the two intermediate parameters used to obtain the two children solutions. It is to be noted that the calculated numerical value of the parameter val is found to be always either greater than or equal to 0.5.

Moreover, during the initial stages of an evolution where the mating solutions are more likely to be far from each other, the obtained value of the parameter val is found to be on the higher side and consequently, the exploration phenomenon of an RGA is promoted during these stages. On the other hand, the measured values for the parameter val are seen to have the lower values (close to 0.5), when the mating parents are found to be relatively close to one another during the later stages of evolution, and due to this reason, the algorithm is able to converge after exploiting the optimum solution.

Boundary constraint handling technique: If a child solution is seen to exceed the upper boundary of the variable (V_u^i) or lie below the lower boundary of the variable (V_l^i), then it is assigned a value equal to V_u^i or V_l^i , respectively.

Child identification conditions: This step is used to identify or recognize the two offsprings (c_1 and c_2) as the first and second child of the created pair of new solutions. When a random number, produced in the range of 0

and 1, is found to have a value either greater than or equal to 0.5, c_2 and c_1 are identified as the first and second children of the yielded pair of solutions. On the other hand, c_1 and c_2 are considered as the first and second members of the generated pair of offsprings, whenever a random number (ranging from 0 to 1) is found to be less than 0.5. These conditions are named as child identification conditions. This is done to incorporate diversity in the population.

A pseudo-code of the proposed DX operator is provided in Fig. 2. Moreover, the basic mechanism of DX is illustrated in Fig. 3 in a two-dimensional variable space. When the direction probability (p_d) is taken to be equal to 1.0, the children solutions follow the current best solution of the population. On the other hand, the offspring are observed to go far from the current best solution, when p_d is found to be equal to 0.0. However, with a value of p_d lying between (0, 1), the children solutions may or may not follow the current best solution.

3.2 Settings and influence of the parameters of DX

There are four user-defined parameters in the proposed directional crossover operator, such as crossover probability (p_c), variable-wise crossover probability (p_{cv}), multiplying factor (α), and directional probability (p_d).

Crossover probability (p_c): Theoretically, p_c can vary from 0 to 1. However, the selection of a suitable value for this parameter depends on the nature of the objective function.

Variable-wise crossover probability (p_{cv}): Similarly to p_c , this parameter has also the range of (0, 1). However, finding a suitable value for it is again seen to be dependent on the nature of the objective function.

Multiplying factor (α): This is the one of the factors to control the distances among children solutions and the mating parents. Theoretically, α can be set to a value from zero to a large positive number. However, the best working range for this parameter is found to be from 0.5 to 2.5. Furthermore, it is recommended to assign values less than or equal to 1.0 and greater than or equal to 1.0 to solve relatively simple and complex optimization problems, respectively. It is to be noted that with the increase in the value of α , diversification power of the algorithm is expected to be more, as the children solutions are going to be created far from the parent solutions. On the contrary, intensification capability of an RGA is found to be enhanced, when α is set to a comparatively smaller value, as the children solutions are expected to be generated closer to the parents.

Fig. 2 The pseudo-code of the proposed DX operator

Input: Two mating solutions (p_1 and p_2) with d number of variables, DX parameters
Output: Two children solutions ($Child_1$ and $Child_2$)

```

if rand  $\leq p_c$ 
  for  $i = 1$  to  $d$ 
    if rand  $\leq p_{cv}$ 
      if  $|p_1^i - p_2^i| > 0$ 
        Calculate  $p_{mean}^i$ 
        val and  $\beta \leftarrow$  using Eq. (1) and Eq. (2)
        if  $(p_{best}^i \geq p_{mean}^i)$ 
          if  $r_1 \leq p_d$ 
             $c_1$  and  $c_2 \leftarrow$  use Eq. (3) and Eq. (4)
          else
             $c_1$  and  $c_2 \leftarrow$  use Eq. (5) and Eq. (6)
          endif
        else
          if  $r_1 \leq p_d$ 
             $c_1$  and  $c_2 \leftarrow$  use Eq. (7) and Eq. (8)
          else
             $c_1$  and  $c_2 \leftarrow$  use Eq. (9) and Eq. (10)
          endif
        endif
        Apply boundary constraint technique
        Apply child identification conditions
      else
         $p_{mean}^i = p_1^i$ 
        if  $(p_{best}^i \neq p_{mean}^i)$ 
          val and  $\beta \leftarrow$  using Eq. (11) and Eq. (12)
          if  $r_1 \leq p_d$ 
             $c_1$  and  $c_2 \leftarrow$  use Eq. (13) and Eq. (14)
          else
             $c_1$  and  $c_2 \leftarrow$  use Eq. (15) and Eq. (16)
          endif
          Apply boundary constraint technique
          Apply child identification conditions
        else
           $Child_1^i = p_1^i$ 
           $Child_2^i = p_2^i$ 
        endif
      endif
    else
       $Child_1^i = p_1^i$ 
       $Child_2^i = p_2^i$ 
    endif
  end of for loop
else
   $Child_1^i = p_1^i$ 
   $Child_2^i = p_2^i$ 
endif

```

Directional probability (p_d): As discussed earlier, the obtained directional information may not be always correct to identify the promising areas in the search space, where the better solutions in terms of the fitness values are present. Due to this reason, it is assumed that the collected directional information is correct probabilistically and in general, it has the higher probability value. Although this parameter p_d has the theoretical range of (0, 1), normally, it is set between 0.5 and 1. For the higher value (close to one) of p_d , the exploitation ability of an RGA is expected to be increased, as most of the solutions are directed to follow the best-obtained solution so far.

On the other hand, the exploration power is seen to be more for the lower value (close to 0.5) of p_d , as most of the children solutions are going to be generated in random directions. Therefore, to solve a unimodal and simpler optimization problem, the higher value (close to one) of p_d is recommended, whereas a small value (near to 0.5) should be assigned for this parameter to solve multimodal and difficult functions. It is also to be noted that the appropriate values of the above parameters are decided through some trials experiments carried out in a systematic approach.

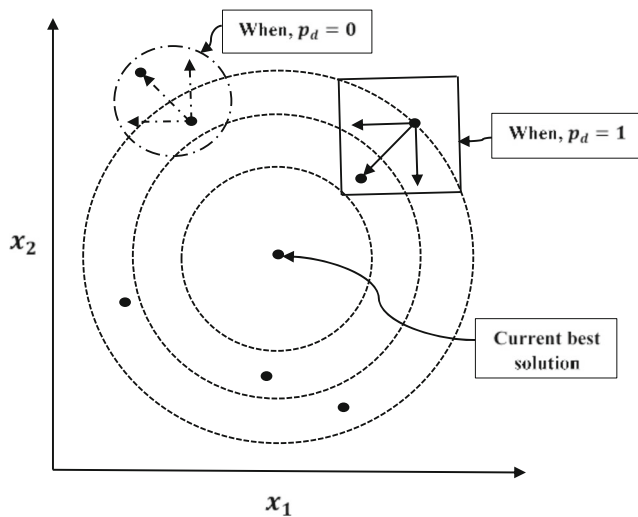


Fig. 3 A schematic view to illustrate the basic mechanism of DX operator in a 2-D variable space

In Fig. 4, the positions of the newly created children solutions (c_1 and c_2) from two mating parents (such as, $p_1 = 2.0$, and $p_2 = -2.0$) with different random numbers (r) have been shown, when (a) $p_d = 1.0$, and (b) $p_d = 0.0$. Here, p_{best} (current best solution's position) has been considered either greater or equal to p_{mean} (mean position of the two parents). Moreover, the upper and lower limits of the variable have been taken as 10 and -10 , respectively, and the value of the multiplying factor (α) has been considered as 1.0. From the figure, it is obvious that the positions of the children solutions are following non-linear exponential distributions (as the equations for generating children solutions involve exponential terms). Furthermore, the maximum distances between Parent 1 (p_1) and Child 1 (c_1) are found to be more than that between Parent 2 (p_2) and Child 2 (c_2) in both the cases. In addition, with the increase of

random numbers from 0 to 1, a converging trend has been seen in the distributions of children solutions (c_1 and c_2) with respect to the mean position of the parent solutions (p_{mean}). This denotes that the distance between two offsprings is found to be more for the low value of random number (near to zero), while for the high value of random number (close to one), this distance is going to be reduced. However, the above mentioned trend of the children's positional distribution is dependent on the value of the multiplying factor (α).

In summary, a few points regarding how the proposed DX operator theoretically helps an RGA to reach the globally optimum solution are presented as follows:

- The proposed crossover operator is influenced by the directional information of the problem, which helps the algorithm to conduct its search in the most potential areas of the variable space.
- The directional probability (p_d) is used to keep a proper balance between exploration and exploitation phenomena of an RGA.
- In addition, a suitable value of the multiplying factor (α) is also responsible to maintain a proper balance between diversification and intensification processes of the algorithm.
- Other parameters, such as p_c and p_{cv} are used to make the search process more efficient probabilistically.
- Furthermore, the intermediate parameter val promotes the exploration capability of the algorithm at the earlier stages of the evolution, while exploitation power is seen to be increased at the later stages.
- The use of random numbers in the proposed DX operator benefits in exploring the search space more efficiently.
- Finally, the developed directional crossover scheme is a parent-centric crossover operator, which utilizes the directional information to yield the children solutions.

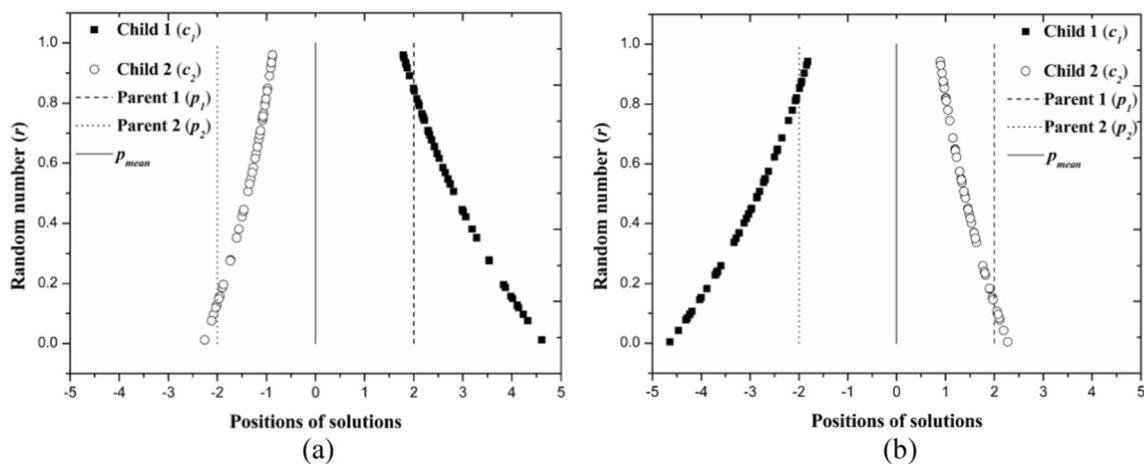


Fig. 4 With the condition $p_{best} \geq p_{mean}$, positions of the children solutions with different random numbers, when (a) $p_d = 1.0$, and (b) $p_d = 0.0$

4 Results and discussion

The performance of a real-coded genetic algorithm (RGA) with the proposed directional crossover operator has been examined through several experiments and the obtained results have been compared to other state-of-the art crossover schemes. The details of these experiments are described as follows:

4.1 The first experiment

In the first experiment, we have chosen six well-known optimization functions with different attributes. The mathematical expressions and the variables’ boundaries of the test functions are provided in Table 1, where d indicates the dimensionality of a problem. Among the functions, three (F01, F02, and F03) are the unimodal type and the rests (F04, F05, and F06) are having the characteristics of multimodality. A unimodal function (refer to Fig. 5a) consists of only one globally optimum basin with no other locally optimum point, while several local optimum basins along with a globally optimum point have been found in the objective space of a multimodal problem (see Fig. 5b). To test the exploitation power and convergence rate of an algorithm, unimodal problems are suitable, whereas multimodal functions are generally used to check the exploration ability of an algorithm, as it has to overcome the local optima trapping problem [63]. An RGA equipped with a tournament selection scheme having a tournament size equal to 2, the proposed DX operator, polynomial mutation scheme [64], and the replacement operation introduced by Deb [65] has been used in the experiment. Each of the functions with 30 variables (i.e., $d=30$) has been solved for 25 times and the optimum results are recorded for all the runs. From these

obtained results, the best, mean, median, worst and standard deviation are calculated. To compare the results obtained using the proposed DX operator, three other popular crossover operators, such as Simulated Binary Crossover (SBX) [51], Laplace Crossover [59], and Rayleigh Crossover (RX) [61] are also used to solve the test functions in a similar kind of RGA’s framework, as in case of the previous one except for the crossover scheme. In addition, a parallel-structured RGA, namely RGA-RDD [45] has also been applied to solve the selected functions. In this RGA-RDD, a ranking selection, direction-based crossover (DBX), dynamic random mutation (DRM), and a generational replacement (GR) with an elitism strategy have been used as the stochastic operators.

4.1.1 Parameters’ settings

The values of the parameters for all the RGAs are chosen after some trial experiments. The common parameters for the first four RGAs are kept the same and these are as follows: population size ($N = 5 \times d$), maximum number of generations ($\max_gen = 1500$), crossover probability ($p_c = 0.9$), mutation probability ($p_m = \frac{1}{d}$), and distribution index for the polynomial mutation ($\eta_m = 20$). Other algorithm-specific parameters are as follows:

- RGA-DX: $p_{cv} = 0.9$, $\alpha = 0.95$, and $p_d = 0.75$.
- RGA-SBX: Distribution index for SBX ($\eta_c = 1$).
- RGA-LX: Location parameter ($a = 0$) and scale parameter ($b = 0.2$).
- RGA-RX: Scale parameter ($s = 1$).
- RGA-RDD: Population size ($N = 5 \times d$), maximum number of generations ($\max_gen = 1500$), proportional parameter for ranking selection ($p = 0.1$), crossover

Table 1 A set of six test functions with their mathematical expressions

Name of the function	Mathematical expression	Variable bounds
F01: Sphere	$f(x) = \sum_{i=1}^d x_i^2$	$[-100,100]^d$
F02: Sum of different powers	$f(x) = \sum_{i=1}^d x_i ^{i+1}$	$[-100,100]^d$
F03: Rotated hyper-ellipsoid	$f(x) = \sum_{i=1}^d \sum_{j=1}^i x_j^2$	$[-65, 65]^d$
F04: Griewank	$f(x) = \frac{\sum_{i=1}^d x_i^2}{4000} - \prod_{i=1}^d \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^d$
F05: Rastrigin	$f(x) = 10d + \sum_{i=1}^d [x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]^d$
F06: Levy	$f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 [1 + 10\sin^2(\pi w_i + 1)] + (w_d - 1)^2 [1 + 10\sin^2(\pi w_d)]$, where $w_i = 1 + (x_i - 1)/4$	$[-5.12, 5.12]^d$

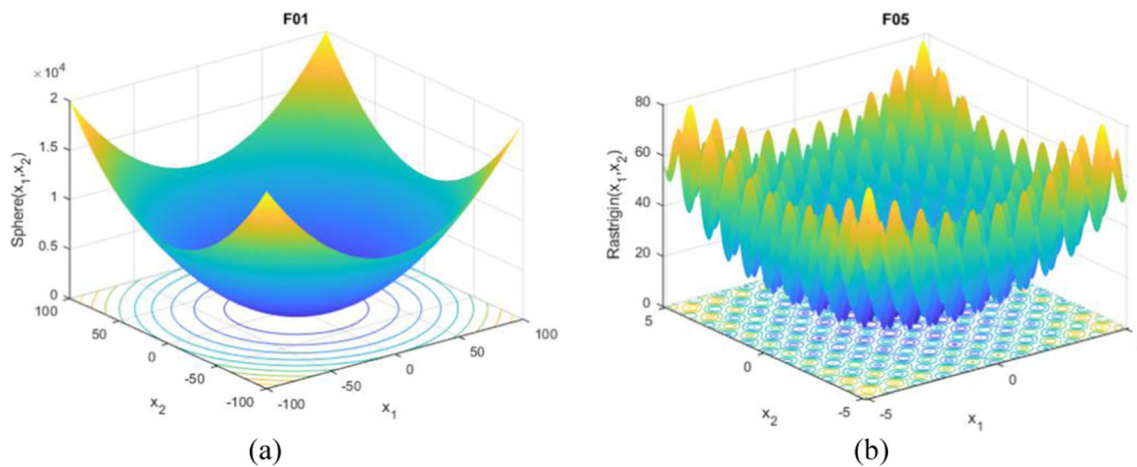


Fig. 5 Plots of objective functions with two variables: **a** Sphere (F01), and **b** Rastrigin (F05)

probability ($\lambda = 0.1$), random perturbation factor ($\phi_0 = 0.5$), parameter to control decay rate of mutation step size ($b = 1$), and a very small positive number ($\varepsilon = 1e^{-300}$).

An algorithm stops when it reaches the maximum number of generations (max_gen). The selected parameters are found to yield the better results in most of the cases during the present experiments. However, the authors are not claiming the fact that these are the best optimal parameters for the algorithms to perform in all the cases. It is to be noted that, in all the experiments, we have used these selected values of parameters as mentioned above. Therefore, whenever the dimension (d) of a problem changes, the population size (N) and mutation probability (p_m) also vary accordingly. The obtained results using the five RGAs have been reported in Table 2 (best results are in bold).

From the results, it is easy to understand that the RGA with the proposed DX operator (RGA-DX) has outperformed all other RGAs equipped with other popular crossover mechanisms in this experiment. Furthermore, in Fig. 6, the evolutions of the average current best fitnesses of all the runs have been shown for all five RGAs. From the figure, it is obvious that the RGA with the proposed crossover exhibits the faster convergence rate compared to others, as it is able to reach near the globally optimum solution in less number of generations. In addition, the trajectories of the average values of the 1st variable in the population have been given in Fig. 7 for all six test functions to observe the exploration and exploitation phenomena during the evolutions.

In a similar way, the five RGAs are also applied to solve the same six benchmark functions with the increased number of variables (d), such as 60, 90, and 120. This is done to verify and compare the performance of RGA-DX in the larger variable space with that of the other ones. The parameters for all the RGAs are taken in a similar fashion, as mentioned in Section 4.1.1. The obtained results are provided in Tables 3,

4, and 5, respectively. In these cases also, the RGA-DX is able to yield the better results compared to that of the other RGAs.

4.1.2 Exploration analysis

It is debated [63] that multimodal optimization problems, where there exist several locally optimal basins along with the global one, are preferable to examine the exploration ability of an optimization method. In our experiment, three functions, namely Griewank (F04), Rastrigin (F05), and Levy (F06) are multimodal functions and these benchmark functions have been tested on varying numbers of variables, where the difficulty levels to solve the problems increase with the increase of dimensions of the functions. The results given in Tables 2, 3, 4, and 5 establish the fact that the RGA with the proposed DX operator has the better exploration capability compared to others. The proposed crossover mechanism has been designed so wisely that the algorithm may able to explore the variable space in a better way. During the earlier stages of an evolution, the calculated parameter val , multiplying factor (α), and the child identification conditions help to spread the search process in the wider regions of the variable space and due to which, the search process becomes more diversified.

4.1.3 Exploitation analysis

The unimodal functions, such as Sphere (F01), Sum of different powers (F02), and Rotated hyper-ellipsoid (F03) are seen to have one global basin with no other locally optimum points and therefore, these are recommended for testing the exploitation power of an optimization algorithm [63]. The yielded results (refer to Tables 2, 3, 4, and 5) confirm that the RGA with the developed DX operator has superior exploitation or intensification strength compared to the other ones. The use of directional information obtained from the

Table 2 Comparison of results for the test functions (F01-F06) with 30 dimensions

Function		RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
F01	Best	3.447E-39	3.700E-10	2.114E-04	1.393E-21	6.815E-31
	Mean	1.012E-37	1.143E-09	1.112E-03	1.782E-20	3.596E-29
	Median	6.516E-38	9.543E-10	9.944E-04	1.078E-20	1.121E-29
	Worst	3.499E-37	3.370E-09	4.056E-03	9.960E-20	2.715E-28
	SD	8.946E-38	7.111E-10	7.704E-04	2.121E-20	6.532E-29
F02	Best	2.677E-56	1.717E-17	5.482E-07	1.035E-46	9.721E-14
	Mean	9.894E-49	1.463E-14	8.330E-05	4.195E-40	1.664E+02
	Median	2.171E-51	2.848E-15	3.817E-05	1.256E-43	2.611E+01
	Worst	9.819E-48	1.507E-13	3.672E-04	8.248E-39	7.844E+02
	SD	2.349E-48	3.183E-14	1.141E-04	1.654E-39	2.523E+02
F03	Best	8.082E-38	2.378E-10	1.493E-03	2.952E-21	1.600E-16
	Mean	1.514E-36	6.867E-09	5.804E-03	9.239E-20	4.371E-12
	Median	3.731E-37	5.366E-09	4.805E-03	5.986E-20	4.154E-13
	Worst	2.473E-35	2.309E-08	1.370E-02	3.552E-19	2.788E-11
	SD	4.868E-36	5.077E-09	3.114E-03	8.713E-20	8.218E-12
F04	Best	0.000E+00	5.780E-10	8.472E-04	0.000E+00	1.110E-16
	Mean	2.958E-04	8.076E-03	1.248E-02	3.841E-03	1.604E-02
	Median	0.000E+00	7.396E-03	1.243E-02	0.000E+00	1.478E-02
	Worst	7.396E-03	3.691E-02	4.076E-02	2.952E-02	4.671E-02
	SD	1.479E-03	9.621E-03	1.123E-02	7.554E-03	1.469E-02
F05	Best	0.000E+00	5.295E-09	6.284E-04	0.000E+00	2.985E+01
	Mean	0.000E+00	1.856E-07	3.162E-03	6.088E-06	9.471E+01
	Median	0.000E+00	6.207E-08	2.037E-03	4.218E-11	6.567E+01
	Worst	0.000E+00	7.728E-07	1.078E-02	1.521E-04	2.900E+02
	SD	0.000E+00	2.499E-07	2.795E-03	3.043E-05	7.384E+01
F06	Best	1.500E-32	2.674E-13	1.956E-07	2.465E-24	1.056E-26
	Mean	1.500E-32	1.368E-12	1.114E-06	4.405E-12	1.074E-02
	Median	1.500E-32	8.472E-13	9.397E-07	2.561E-23	6.084E-23
	Worst	1.500E-32	3.581E-12	2.583E-06	1.101E-10	8.953E-02
	SD	0.000E+00	9.979E-13	6.682E-07	2.203E-11	2.969E-02

Best results are marked in bold

problem dynamically, variable-wise crossover probability (p_{cv}) and the higher directional probability ($p_d > 0.5$) influence the search process to intensify in the most promising areas of the variable space and thus, promote the exploitation phenomenon in the later stages of an evolution.

4.1.4 Convergence analysis

During the initial steps of an evolution, abrupt modulations in the values of the search agents are usually observed and gradually, these modulations are found to be reduced over the generations [66]. This is due to the fact

that the exploring factor of an optimization approach is seen to be dominating at the beginning of an evolution, whereas the algorithm is found to exploit the solutions to reach the globally optimum point afterward. In Fig. 7, the evolutions of the average values of the 1st variable in the population have been depicted. From the figure, it is easy to comprehend that during the initial generations, changes in the average values of the 1st variable are more unpredictable, and this highlights the state, where the exploration power of the algorithm is found to be dominating in nature. However, in the later stages, these changes are gradually reduced and the algorithm is seen to converge to a globally optimum solution. This characteristic

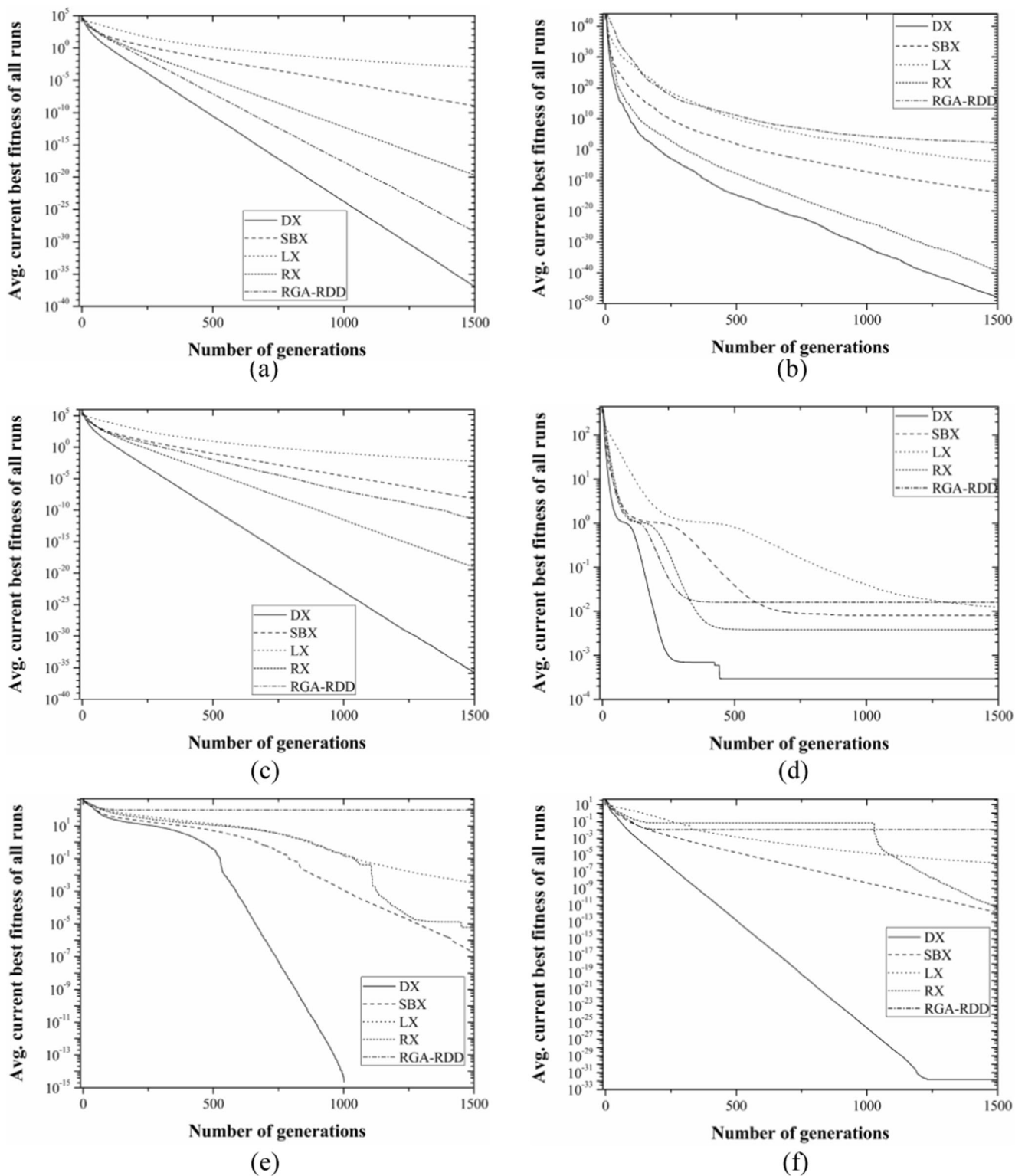


Fig. 6 Evolutions of average current best fitness values of all the runs using RGAs with five different crossover operators for the functions: **a** F01, **b** F02, **c** F03, **d** F04, **e** F05, and **f** F06

of the RGA with the proposed DX is matching with the convergence criterion mentioned in [66] and hence, it can be claimed that the RGA-DX converges to an optimum solution in the variable space.

4.1.5 Statistical analysis

To analyze the obtained results statistically, pairwise and multiple comparisons have been performed. These tests are non-

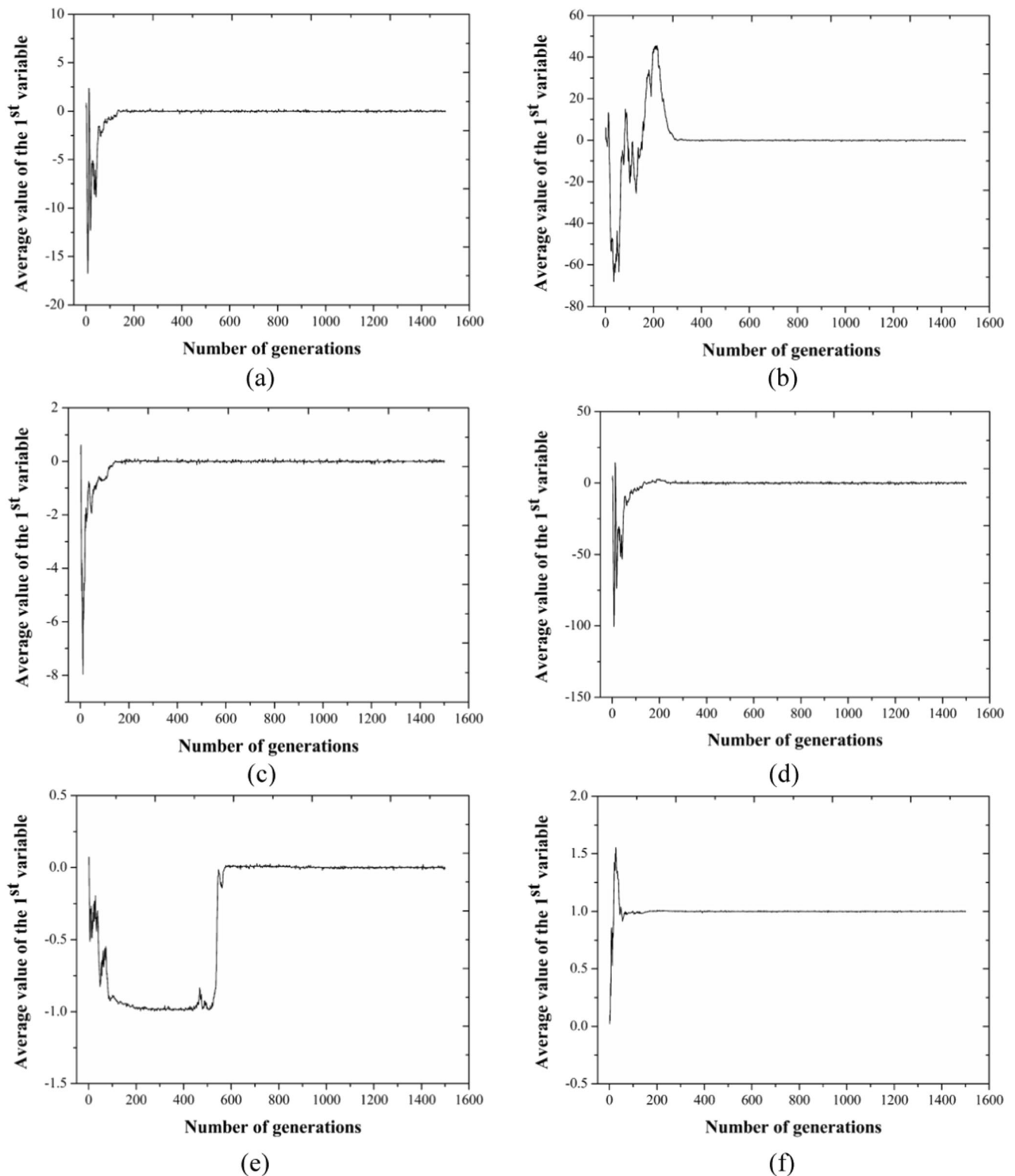


Fig. 7 Evolutions of the average value of the 1st variable in the population for the functions: **a** F01, **b** F02, **c** F03, **d** F04, **e** F05, and **f** F06

parametric in nature and the mean results provided in Tables 2, 3, 4, and 5 are considered as the raw data, on which the present analyses have been carried out. For the pairwise comparisons, two separate tests, namely Sign test and Wilcoxon's test [67],

have been considered. On the other hand, Friedman's test, aligned Friedman's test and Quade test with four post-hoc procedures, such as Holland, Rom, Finner and Li [67, 68], have been selected for the multiple comparisons.

Table 3 Comparison of results for the test functions (F01-F06) with 60 dimensions

Function		RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
F01	Best	8.937E-22	1.935E-04	2.892E-01	4.951E-08	3.634E-15
	Mean	1.832E-21	4.386E-04	5.182E-01	1.435E-07	2.315E-14
	Median	1.472E-21	4.285E-04	5.125E-01	1.368E-07	2.102E-14
	Worst	4.128E-21	9.009E-04	8.872E-01	2.915E-07	5.913E-14
	SD	8.990E-22	1.473E-04	1.537E-01	5.979E-08	1.229E-14
F02	Best	7.859E-38	1.833E+03	3.394E+25	2.216E-14	1.495E+08
	Mean	1.248E-29	1.011E+06	3.150E+33	2.712E-09	4.954E+18
	Median	4.138E-32	1.298E+05	3.897E+29	2.067E-10	9.497E+13
	Worst	1.421E-28	1.065E+07	4.422E+34	5.699E-08	1.236E+20
	SD	3.469E-29	2.244E+06	9.792E+33	1.133E-08	2.472E+19
F03	Best	9.123E-21	1.684E-03	2.623E+00	7.781E-07	2.194E-05
	Mean	2.596E-20	4.168E-03	7.147E+00	1.574E-06	1.394E-03
	Median	2.043E-20	4.120E-03	7.314E+00	1.402E-06	6.719E-04
	Worst	5.527E-20	9.122E-03	1.164E+01	3.559E-06	5.592E-03
	SD	1.338E-20	1.545E-03	2.142E+00	7.429E-07	1.813E-03
F04	Best	0.000E+00	2.771E-04	2.121E-01	5.130E-08	1.478E-09
	Mean	0.000E+00	4.299E-03	3.487E-01	1.084E-03	3.943E-03
	Median	0.000E+00	5.749E-04	3.356E-01	1.615E-07	4.882E-09
	Worst	0.000E+00	1.505E-02	5.919E-01	1.232E-02	1.477E-02
	SD	0.000E+00	5.079E-03	9.041E-02	3.107E-03	5.227E-03
F05	Best	3.752E-12	1.600E-03	7.718E+00	7.543E+00	7.064E+01
	Mean	9.661E-07	3.928E-01	1.453E+01	1.576E+01	1.518E+02
	Median	3.927E-09	3.016E-02	1.247E+01	1.514E+01	1.174E+02
	Worst	2.081E-05	4.527E+00	2.445E+01	2.497E+01	5.604E+02
	SD	4.145E-06	1.031E+00	4.449E+00	4.705E+00	1.237E+02
F06	Best	2.044E-23	2.671E-07	1.997E-04	1.251E-10	1.219E-14
	Mean	6.456E-23	4.833E-07	6.889E-04	5.626E-10	2.507E-02
	Median	5.687E-23	4.141E-07	7.262E-04	5.443E-10	1.920E-13
	Worst	1.194E-22	9.095E-07	1.432E-03	1.481E-09	8.953E-02
	SD	2.903E-23	1.921E-07	2.917E-04	3.436E-10	4.103E-02

Best results are marked in bold

The results of the pairwise comparisons between the RGA with the proposed DX operator and other RGAs have been reported in Table 6, where the first row contains the number of wins and losses of the proposed approach compared to the others. An optimization

technique is assumed to be the winner, if the average result of the total runs is seen to be better than the other one. As we have a total of 24 cases for comparisons, an algorithm should win at least in 18 cases, so that it can be declared as the overall winner compared

Table 4 Comparison of results for the test functions (F01-F06) with 90 dimensions

Function		RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
F01	Best	2.738E-14	4.308E-02	1.171E+01	2.055E-03	2.172E-09
	Mean	5.272E-14	5.967E-02	2.076E+01	4.130E-03	5.204E-09
	Median	5.030E-14	5.543E-02	2.035E+01	3.820E-03	4.521E-09
	Worst	8.763E-14	7.912E-02	3.165E+01	7.400E-03	1.146E-08
	SD	1.239E-14	1.232E-02	5.839E+00	1.247E-03	2.285E-09
F02	Best	8.311E-16	3.439E+23	7.661E+68	7.248E+09	5.575E+29
	Mean	1.232E-08	4.153E+33	4.424E+83	2.675E+15	2.396E+42
	Median	5.705E-12	1.751E+28	2.780E+79	1.671E+12	1.707E+38
	Worst	2.071E-07	1.035E+35	6.541E+84	2.432E+16	3.823E+43
	SD	4.282E-08	2.070E+34	1.496E+84	6.881E+15	7.983E+42
F03	Best	3.490E-13	5.256E-01	1.701E+02	3.293E-02	3.703E-02
	Mean	8.046E-13	8.578E-01	2.995E+02	6.032E-02	4.749E-01
	Median	7.776E-13	8.239E-01	2.733E+02	5.647E-02	4.315E-01
	Worst	1.445E-12	1.441E+00	4.974E+02	1.100E-01	1.798E+00
	SD	2.925E-13	2.328E-01	9.303E+01	1.950E-02	4.012E-01
F04	Best	1.288E-14	2.072E-02	1.107E+00	1.390E-03	3.708E-06
	Mean	3.220E-14	3.652E-02	1.187E+00	3.782E-03	2.870E-03
	Median	2.787E-14	3.385E-02	1.183E+00	2.384E-03	1.284E-05
	Worst	6.062E-14	5.488E-02	1.285E+00	1.133E-02	1.233E-02
	SD	1.388E-14	1.033E-02	5.225E-02	3.172E-03	4.367E-03
F05	Best	7.865E-04	2.535E-01	4.647E+01	4.114E+01	1.224E+02
	Mean	2.610E+00	1.017E+01	6.416E+01	5.898E+01	3.066E+02
	Median	6.017E-02	7.111E+00	6.343E+01	6.010E+01	1.871E+02
	Worst	1.172E+01	2.939E+01	8.038E+01	7.251E+01	8.538E+02
	SD	4.131E+00	8.444E+00	8.567E+00	7.351E+00	2.604E+02
F06	Best	7.728E-16	2.208E-05	1.204E-02	9.838E-06	4.645E-11
	Mean	1.492E-15	5.737E-05	2.473E-02	3.303E-01	4.655E-02
	Median	1.262E-15	6.075E-05	2.308E-02	2.020E-05	1.667E-09
	Worst	4.051E-15	7.787E-05	4.376E-02	5.856E+00	2.686E-01
	SD	7.239E-16	1.341E-05	8.062E-03	1.247E+00	7.805E-02

Best results are marked in bold

Table 5 Comparison of results for the test functions (F01-F06) with 120 dimensions

Function		RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
F01	Best	5.710E-10	3.948E-01	9.647E+01	2.458E-01	1.330E-06
	Mean	9.379E-10	7.045E-01	1.698E+02	4.878E-01	2.742E-06
	Median	9.532E-10	7.049E-01	1.685E+02	4.765E-01	2.548E-06
	Worst	1.435E-09	8.856E-01	2.571E+02	7.319E-01	5.548E-06
	SD	2.355E-10	1.265E-01	3.986E+01	1.281E-01	1.008E-06
F02	Best	1.121E+03	2.108E+48	1.025E+124	7.078E+38	4.355E+75
	Mean	5.995E+08	6.402E+64	7.005E+137	2.970E+49	6.617E+90
	Median	3.962E+06	4.706E+58	1.012E+130	4.111E+44	3.131E+83
	Worst	6.658E+09	1.218E+66	1.750E+139	6.903E+50	1.654E+92
	SD	1.728E+09	2.497E+65	3.501E+138	1.378E+50	3.307E+91
F03	Best	1.080E-08	1.069E+01	1.860E+03	4.995E+00	9.599E-01
	Mean	1.752E-08	1.384E+01	4.148E+03	8.738E+00	6.769E+00
	Median	1.760E-08	1.365E+01	4.054E+03	8.894E+00	3.918E+00
	Worst	2.729E-08	1.807E+01	6.933E+03	1.182E+01	3.797E+01
	SD	3.847E-09	2.086E+00	1.345E+03	1.898E+00	7.747E+00
F04	Best	2.589E-10	2.043E-01	1.868E+00	9.750E-02	3.384E-04
	Mean	4.096E-10	2.806E-01	2.528E+00	1.876E-01	3.973E-03
	Median	4.205E-10	2.797E-01	2.517E+00	1.851E-01	5.762E-04
	Worst	6.046E-10	3.551E-01	3.314E+00	2.641E-01	1.788E-02
	SD	9.245E-11	4.432E-02	3.587E-01	4.751E-02	5.935E-03
F05	Best	8.146E+00	1.675E+00	1.151E+02	9.592E+01	1.423E+02
	Mean	2.648E+01	2.851E+01	1.481E+02	1.227E+02	2.768E+02
	Median	2.739E+01	2.799E+01	1.515E+02	1.242E+02	1.960E+02
	Worst	4.997E+01	5.594E+01	1.713E+02	1.527E+02	1.167E+03
	SD	1.171E+01	1.429E+01	1.771E+01	1.502E+01	2.640E+02
F06	Best	7.932E-12	4.400E-04	1.136E-01	1.002E-03	1.539E-08
	Mean	1.772E-11	6.495E-04	1.816E-01	4.876E-01	7.520E-02
	Median	1.692E-11	6.460E-04	1.682E-01	2.536E-03	8.953E-02
	Worst	2.620E-11	8.028E-04	2.937E-01	3.476E+00	2.686E-01
	SD	4.465E-12	9.860E-05	5.121E-02	1.063E+00	8.833E-02

Best results are marked in bold

to the other one with a significance level equal to 0.05 (refer to Table 4 of [67]). Therefore, the proposed RGA-DX is observed to be the winner in all the pairwise comparisons. Furthermore, p -values obtained using the Sign test are found to be less than 0.05. It means that there has been a significant difference in

performance between the RGA-DX and others. It is observed that all the p -values of the Sign test are identical in nature. It may be due to the fact that in all the pairwise comparisons, the RGA-DX has won with the considerable differences for the 24 cases. Similar findings have also been observed during the pairwise

Table 6 Pairwise comparison results: RGA-DX vs. other RGAs

RGA-DX vs.	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
Wins/Losses	24/0 ^a	24/0 ^a	24/0 ^a	24/0 ^a
Sign test <i>p</i> -value	1.19E-07	1.19E-07	1.19E-07	1.19E-07
Wilcoxon <i>p</i> -value	1.82E-05	1.82E-05	1.82E-05	1.82E-05

^a Level of significance = 0.05

comparisons of results using the Wilcoxon's test. These analyses ensure that the performance of the proposed RGA-DX is significantly better than that of the others.

In connection to the multiple comparisons, the statistical tests have been carried out using a CONTROLTEST java package [67], downloaded from the SCI2S website (<http://sci2s.ugr.es/sicidm>). The relative ranks of the RGAs according to the Friedman, Aligned Friedman and Quade tests are given in Table 7. A lower rank indicates a superior in performance compared to others. For all of the three tests, the RGA-DX has achieved the lowest rank. This implies that it is the best performer among all the RGAs considered in this experiment. In addition, the statistics and *p*-values of the tests have been provided in the last two rows of Table 7, respectively. The statistics for both the Friedman and the Aligned Friedman are distributed following the chi-square distribution with 4 degrees of freedom. In case of the Quade test, this distribution is found to be in line with the F-distribution with 4 and 92 degrees of freedom. Moreover, the *p*-values for all the three tests are observed to be less than 0.05. This shows that the RGA with the proposed DX technique is performing significantly better compared to the other ones.

Table 8 provides the results of the contrast estimation, which is supposed to quantify the difference in performance between two algorithms. This method is considered as the safer metric in performance comparisons between the algorithms and this can be helpful to determine, how much an algorithm is superior to another one [67]. Each row of Table 8 gives the comparison results between the corresponding approach of the 1st column and rest of the methods. If the estimated quantity is found to be positive, the algorithm present in the 1st column is assumed to be a better one and the

Table 7 Friedman, Aligned Friedman and Quade ranks of the RGAs

Algorithm	Friedman	Aligned Friedman	Quade
RGA-DX	1.000	44.271	1.000
RGA-SBX	3.167	46.792	3.127
RGA-LX	4.500	91.792	4.567
RGA-RX	2.792	53.542	2.697
RGA-RDD	3.542	66.104	3.610
Statistic	63.500	17.932	26.010
<i>p</i> -value	4.83E-11	0.00127	1.96E-14

Table 8 Contrast estimation results of the first experiment

	RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
RGA-DX	0	0.04702	1.311	0.2129	0.3492
RGA-SBX	-0.04702	0	1.264	0.1658	0.3022
RGA-LX	-1.311	-1.264	0	-1.098	-0.9614
RGA-RX	-0.2129	-0.1658	1.098	0	0.1364
RGA-RDD	-0.3492	-0.3022	0.9614	-0.1364	0

Best results are marked in bold

reverse is true, when the result is observed to be a negative one. In addition, the more the estimated value, the more will be the difference in performance between the algorithms. For RGA-DX, all the estimated values are found to be positive and this implies that the RGA-DX outperforms all other RGAs participated in the experiments.

Furthermore, to deduce the significant differences between the RGA-DX and others, *z*-values, unadjusted *p*-values and adjusted *p*-values using four post-hoc procedures of the Friedman, Aligned Friedman, and Quade tests have been reported in Table 9. The proposed RGA-DX technique has been considered as the control method for this comparison. Utilizing corresponding *z*-value on a normal distribution $N(0,1)$ [67], the unadjusted *p*-value is obtained for a test. The unadjusted *p*-values (refer to the 5th column of Table 9) for the Friedman and the Quade test indicate significant differences between the RGA-DX and other RGAs, whereas the unadjusted *p*-values yielded using the Aligned Friedman test claim that the RGA-DX is significantly better than the RGA-LX and RGA-RDD only.

Due to some family error accumulation in the process of obtaining unadjusted *p*-values, adjusted *p*-values obtained through post-hoc procedures are recommended to be used in the analysis. Among several techniques, we have chosen four powerful post-hoc procedures, namely Holland, Rom, Finner and Li [67], and the results are given in Table 9. According to the Friedman and Quade tests, RGA-DX is able to produce the significantly better results compared to the other RGAs, while it shows a significant difference in performances only with the RGA-LX at the significance level equal to 0.05, according to the Aligned Friedman test.

Furthermore, a non-parametric convergence performance analysis has been carried out using the Page's trend test [69]. This method had been proposed to test the convergence performance of evolutionary algorithms pairwise and it assumes that an algorithm with the better convergence rate should move faster towards the optimum solution compared to that of another algorithm having the worse convergence performance. This method is supposed to consider the differences between the objective function values of the two evolutionary approaches at various points of the run (cut points) in its calculations. In our case, we have selected 15 cut-points starting

Table 9 Statistical results of post-hoc procedures over all RGAs with RGA-DX as control method at a significance level equal to 0.05

Procedure	<i>i</i>	Algorithm	z-value	Unadjusted <i>p</i> -value	Adjusted <i>p</i> -value			
					<i>P</i> _{Holl}	<i>P</i> _{Rom}	<i>P</i> _{Finn}	<i>P</i> _{Li}
Friedman	1	RGA-LX	7.668116	1.75E-14	6.97E-14	6.66E-14	6.97E-14	1.75E-14
	2	RGA-RDD	5.568513	2.57E-08	7.71E-08	7.71E-08	5.14E-08	2.57E-08
	3	RGA-SBX	4.746929	2.07E-06	4.13E-06	4.13E-06	2.75E-06	2.07E-06
	4	RGA-RX	3.925345	8.66E-05	8.66E-05	8.66E-05	8.66E-05	8.66E-05
Aligned Friedman	1	RGA-LX	4.732406	2.22E-06	8.87E-06	8.46E-06	8.87E-06	1.12E-05
	2	RGA-RDD	2.174293	0.029683	0.086432	0.08905	0.058485	0.130247
	3	RGA-RX	0.923244	0.35588	0.585109	0.71176	0.443725	0.642271
	4	RGA-SBX	0.25104	0.801784	0.801784	0.801784	0.801784	0.801784
Quade	1	RGA-LX	4.833767	1.34E-06	5.36E-06	5.11E-06	5.36E-06	1.37E-06
	2	RGA-RDD	3.537233	4.04E-04	0.001213	0.001213	8.09E-04	4.13E-04
	3	RGA-SBX	2.88219	0.003949	0.007883	0.007898	0.005262	0.00402
	4	RGA-RX	2.299428	0.021481	0.021481	0.021481	0.021481	0.021481

from 100th generation to 1500th generation with a gap of 100. There are two versions of this test, namely the basic version and the alternative version. However, the latter is considered to be more efficient compared to the former [69]. Therefore, the results of the alternative version of the test are summarized in Table 10, where the *p*-values are determined as the probability of rejection of the hypothesis of equal convergence, in support of the alternative hypothesis, i.e., the technique present in the row converges faster than that appeared in the column. The rejected hypotheses (at a significance level of 0.1) are written in bold. From the *p*-values, the following conclusions can be made: RGA-DX has the best convergence performance, followed by RGA-RDD, RGA-RX, and RGA-SBX. RGA-LX is found to have the worst convergence performance in this study.

4.2 The second experiment

It is argued that it incorporates more difficulty to solve an optimization problem (especially, for the multi-modal functions), when the initial population does not bracket the

globally optimum point [70]. Based on this idea, the second experiment has been performed with an initial population away from the actual globally optimum solution for all the six test functions mentioned in Table 1. Here, we have created the initial solutions in the range of (−5,−10) for all the test functions to ensure that the initial population should not cover the global optimum. These six test functions with 30 variables have been solved using five RGAs and the parameters’ settings are kept the same, as mentioned in Section 4.1.1. It is to be noted that the variable boundary constraint handling techniques have not been applied to the algorithms in this experiment. All the functions have been solved by each algorithm 25 times and the average of the obtained best results are reported in Table 11 (best results are marked in bold). From the results, it is clear that the RGA-DX has outperformed the other four RGAs for all the test functions in this experiment.

4.3 The third experiment

To see the sole effect of the crossover operation, we have excluded the mutation operation in the third experiment and redone the 1st experiment with only 30 dimensions of the test functions. Moreover, in this experiment, we have excluded RGA-RDD algorithm, as the mutation operation is attached with the directed crossover operation in this case. The parameters’ settings have been considered the same as narrated in Section 4.1.1 except the mutation probability (*p_m*), which is kept equal to 0.0. The average values of the optimum results obtained in 25 runs have been given in Table 12. In this experiment also, the RGA-DX has retained its excellent performance for all six test functions except for the case of F05, where RGA-SBX has

Table 10 Convergence results (*p*-values) of the first experiment

	RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
RGA-DX	–	0	0	0	0
RGA-SBX	1	–	0	0.999897	1
RGA-LX	1	1	–	1	1
RGA-RX	1	0.000106	0.000008	–	0.993395
RGA-RDD	1	0	0	0.006726	–

Best results are marked in bold

Table 11 Comparison of average results of 25 runs for the test functions (F01-F06) with different variable boundaries

Function	RGA-DX	RGA-SBX	RGA-LX	RGA-RX	RGA-RDD
F01	4.298E-39	1.453E-10	9.058E-06	9.696E-22	6.164E-31
F02	1.452E-55	1.316E-16	5.748E-08	1.920E-41	1.289E-15
F03	6.119E-38	2.106E-09	1.319E-04	1.519E-20	6.161E-13
F04	2.958E-04	9.860E-04	8.759E-03	1.478E-03	4.731E-03
F05	0.000E+00	3.066E-05	5.750E-03	4.633E-04	1.056E+02
F06	1.425E-31	8.669E-11	1.607E+00	1.536E-19	2.051E+00

Best results are marked in bold

been able to yield a slightly better result compared to the RGA-DX.

4.4 Performance comparisons of RGA-DX with other recently proposed optimization algorithms

To check the state-of-the-art, the performance of the proposed RGA-DX has also been compared to that of five recently proposed optimization algorithms, such as Cricket Algorithm (CA) [28], Grasshopper Optimization Algorithm (GOA) [29], Social Spider Optimization (SSO) [31], Sine Cosine Algorithm (SCA) [30], and Search Group Algorithm (SGA) [32]. This experiment has been carried out on the six benchmark functions (refer to Table 1) in a similar way, as in case of the 1st experiment conducted with 30 dimensions ($d = 30$). Therefore, the common parameters, such as the maximum number of generations ($\max_gen = 1500$) and population size ($N = 5 \times d = 150$) are kept the same as mentioned in Section 4.1.1. The algorithm-specific parameters for the optimization techniques are set through some trial experiments and these are as follows:

RGA-DX: the same as those mentioned in Section 4.1.1.

Table 12 Comparison of average results of 25 runs for the test functions (F01-F06) without mutation operation

Function	RGA-DX	RGA-SBX	RGA-LX	RGA-RX
F01	1.36E-87	4.64E-21	1.25E+04	2.21E-52
F02	5.62E-96	2.30E-12	7.67E+36	1.71E-102
F03	4.37E-87	2.17E-20	6.91E+04	2.19E-51
F04	5.92E-04	8.96E-03	1.13E+02	3.55E-03
F05	2.70E+01	2.44E+01	1.19E+02	5.41E+01
F06	1.50E-32	5.23E-24	5.68E+00	5.45E-02

Best results are marked in bold

CA: minimum frequency ($Q_{min} = 0$), $\beta_{min} = 0.2$.

GOA: Maximum and minimum values for the coefficient to shrink the comfort, repulsion, and attraction zones are 1 and 0.00004, respectively.

SSO: Lower and upper female percent factors are taken to be equal to 0.65 and 0.9, respectively.

SCA: The value of the constant to determine the range of the sine and cosine has been considered to be equal to 2.

SGA: Initial value for perturbation factor, minimum value of the perturbation factor, global iteration ratio, search group ratio, and the number of mutated individuals of the search group are taken to be equal to 2, 0.01, 0.3, 0.1, and 5, respectively.

Table 13 shows the average of the obtained best fitness results of 25 runs yielded using RGA-DX and the other five algorithms for each of the test functions (best results are written in bold). Here, it is to be noted that the results obtained using the RGA-DX (refer to Table 13) are the same, with those of the 1st experiment carried out with 30 dimensions (see Table 2). This is due to the fact that the parameters used for the RGA-DX are kept the same in both the experiments. From the results, it can be concluded that the proposed RGA-DX has outperformed all other recent optimization algorithms in terms of accuracy in solution. The proper balance between the population diversity and selection pressure along with the use of the directional information helps the proposed RGA-DX to perform in the better way.

4.5 Scalability analysis

We carry out a scale-up study for the proposed RGA-DX with the increasing number of variables (d). Here, we have varied d from 20 to 200 with an increment of 20. The parameters' settings have been taken in a similar way, as mentioned in Section 4.1.1 except the stopping criterion of the RGA-DX. The algorithm stops, when it reaches the objective function value equal to 0.1. The required number of function evaluations (f_e) to reach the mentioned solution accuracy for each of the function dimensionality levels has been recorded.

Table 13 Comparisons of average results of 25 runs for the test functions (F01-F06) among the proposed RGA-DX and other recent optimization algorithms

Function	RGA-DX	CA	GOA	SSO	SCA	SGA
F01	1.012E-37	1.385E-03	3.973E-06	5.648E-02	9.844E-10	3.119E-04
F02	9.894E-49	4.601E+34	4.040E+36	5.648E-02	1.257E-01	1.996E-07
F03	1.514E-36	6.579E+02	5.474E-01	8.737E-01	5.118E-10	1.560E-02
F04	2.958E-04	5.699E-03	2.022E-02	1.395E-02	3.067E-02	2.310E-03
F05	0.000E+00	6.579E+02	1.101E+02	4.313E+01	1.917E+00	2.834E+01
F06	1.500E-32	7.909E+00	4.868E+00	2.890E-02	1.843E+00	6.800E-01

Best results are marked in bold

Next, a log-log curve has been plotted for each of the test functions, where the X and Y-axis are represented by the values of $\ln(d)$ and $\ln(fe)$, respectively. With the increase of the variable space, it is a bit difficult to trace the globally optimum solution for an optimization technique [70]. Therefore, the purpose of this scale-up study is to see, whether the RGA-DX is able to either hit or come to near the globally optimum point in a larger variable space. In addition, another aim for this analysis is to get an idea of how the parameter fe is varying with the increasing number of variables (d). Figure 8 depicts the linearly fitted curves between $\ln(d)$ and $\ln(fe)$ along with their slopes for the six benchmark functions. From the figure, it is evident that the RGA-DX is able to reach the mentioned objective function's accuracy level even for a very large variable space for all six test functions. Moreover, the required number of function evaluations (fe) increases polynomially ($O(d^{Slope})$) with d over the whole range of the variables considered in this experiment. From the values of the slopes, it is observed that for function F02, the number of required function evaluations is higher than that of the other ones. This may be due to the inherent nature of the problem (F02), for which it becomes more difficult to solve in a very large variable space.

4.6 Time study

Finally, we have performed a time study to make a comparison among the five RGAs on the aspects of computationally expensiveness and convergence rates. To do this, each algorithm has been run for 25 times to solve a single test function, till the time it reaches the objective function value equal to 0.001. The dimensions of all six test functions have been considered as 30 and the parameters' settings are kept the same as in the case of the first experiment, except the stopping criterion. After each run, the required CPU time and the number of function evaluations have been measured and at the end of 25 runs, the average CPU time

(t_{avg}) in seconds and the average number of function evaluations (fe_{avg}) have been calculated. This experiment has been carried out using MATLAB 2017a software on an Intel Core i5 Processor with 3.20 GHz speed and 16GB RAM under Windows 10 platform. The obtained results are provided in Table 14 for all the test functions solved by the five RGAs (the best results are marked in bold).

It is to be noted that, for a certain run, if an algorithm is not able to reach the mentioned function value up to a maximum number of function evaluations equal to $3E5$, we have marked the values of t_{avg} and fe_{avg} for that particular function solved by that algorithm as not available (NA) in Table 14. From the comparisons, it is easy to understand that the RGA-DX requires minimum CPU time and number of function evaluations to achieve the desired objective value compared to the other ones. This observation confirms that the RGA-DX has the fastest convergence rate among the five RGAs.

5 Design problem of cantilever beam

After successfully implementing the RGA with the proposed DX operator to solve a set of six benchmark optimization functions, we have used the RGA-DX to solve a constrained engineering problem. This problem is related to the design of a cantilever beam with discrete rectangular cross-sections (refer to Fig. 9). The objective of this design is to minimize the volume of the said cantilever beam with optimal combinations of the five different cross-sectional areas. The problem contains ten design variables, such as width (b_i) and height (h_i) of each cross-section ($i = 1, \dots, 5$). At the free end of the beam, an external load (P) is applied with the magnitude equal to 50,000 N. Moreover, at the left end of each section, the maximum allowable stress

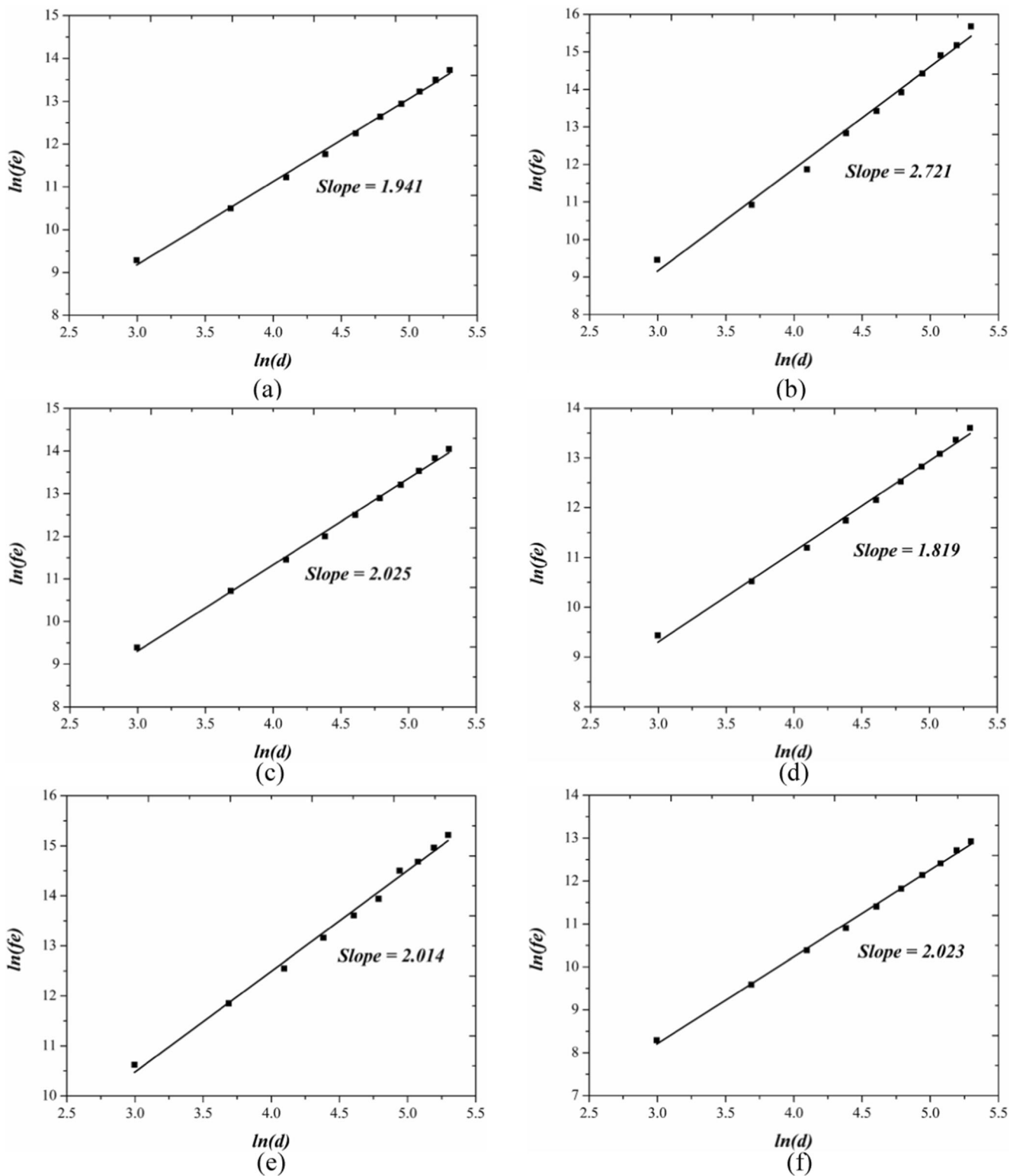


Fig. 8 Scalability study for the functions: a F01, b F02, c F03, d F04, e F05, and f F06

(σ_{max}) is found to be equal to 14,000 N/cm². In addition, the length of each section l_i ($i = 1, \dots, 5$), material elasticity (E) and maximum allowable deflection (Y_{max}) are seen to be equal to 100 cm, 200 GPa and 2.715 cm,

respectively. The height-to-width aspect ratio of each cross-section is limited to less than 20 and there are 11 functional constraints. The problem can be mathematically expressed as follows:

Table 14 Comparisons of the results of the time study

Function	RGA-DX		RGA-SBX		RGA-LX		RGA-RX		RGA-RDD	
	t_{avg} (in Sec.)	fe_{avg}	t_{avg} (in Sec.)	fe_{avg}	t_{avg} (in Sec.)	fe_{avg}	t_{avg} (in Sec.)	fe_{avg}	t_{avg} (in Sec.)	fe_{avg}
F01	0.8041	32,700	1.8857	100,902	3.9594	224,280	1.1716	56,874	2.5138	45,864
F02	1.0146	34,200	2.5502	107,100	4.4469	196,038	1.3968	54,432	14.0907	241,122
F03	0.9477	36,408	2.3431	114,906	5.9038	294,690	1.4142	63,336	3.9501	88,110
F04	0.6021	24,150	1.2725	63,102	2.3350	119,994	0.9239	40,734	1.8959	34,038
F05	2.0698	90,900	2.6857	141,582	4.6862	248,388	3.5038	161,454	NA	NA
F06	0.5969	17,250	1.1043	41,544	2.1708	83,088	0.9364	31,698	NA	NA

Best results are marked in bold

Take, $X = [b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5]^T = [x_1, x_2, \dots, x_{10}]^T$
 Minimize $f(X) = 100(x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10})$
 subject to

$$\begin{aligned}
 g_1(x) &= 10.7143 - \frac{x_1x_2^2}{1000} \leq 0, \\
 g_2(x) &= 8.5714 - \frac{x_3x_4^2}{1000} \leq 0, \\
 g_3(x) &= 6.4286 - \frac{x_5x_6^2}{1000} \leq 0, \\
 g_4(x) &= 4.2957 - \frac{x_7x_8^2}{1000} \leq 0, \\
 g_5(x) &= 2.1429 - \frac{x_9x_{10}^2}{1000} \leq 0, \\
 g_6(x) &= 10^4 \left(\frac{244}{x_1x_2^3} + \frac{148}{x_3x_4^3} + \frac{76}{x_5x_6^3} + \frac{28}{x_7x_8^3} + \frac{4}{x_9x_{10}^3} \right) - 10.86 \leq 0, \\
 g_7(x) &= x_2 - 20x_1 \leq 0, \\
 g_8(x) &= x_4 - 20x_3 \leq 0, \\
 g_9(x) &= x_6 - 20x_5 \leq 0, \\
 g_{10}(x) &= x_8 - 20x_7 \leq 0, \\
 g_{11}(x) &= x_{10} - 20x_9 \leq 0, \\
 \text{and} \\
 1 \leq x_i \leq 5, i = 1, 3, 5, 7, 9, \text{ and } 30 \leq x_j \leq 65, j = 2, 4, 6, 8, 10.
 \end{aligned}
 \tag{17}$$

In the literature, this problem had already been solved using several techniques, such as Sequential Directed Genetic Algorithm Including Memory with Local Search (SDGAM_{INLS}) [60], Mathematical Programming Neural Network (MPNN) [60] and Sequential Unconstrained

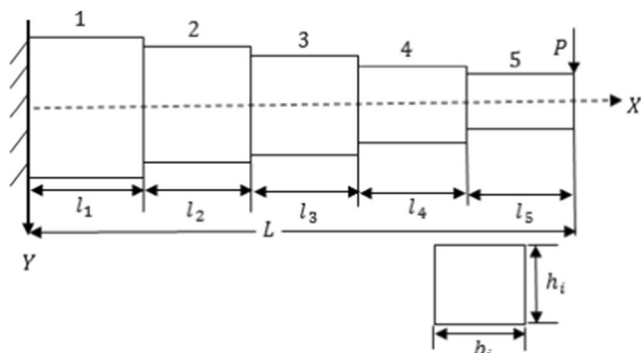


Fig. 9 Schematic view of a cantilever beam with the design parameters

Minimization Techniques (SUMT) [60]. In our case, the problem has been solved using the RGA-DX with a penalty function method [60], where the penalty factor is assigned to be equal to 10^7 . Moreover, it is important to note that the optimum solution obtained using the RGA-DX has the precision of four digits after the decimal point. From the results given in Table 15, it is easy to say that the RGA-DX has been able to generate the best feasible results compared to the others. In addition, contents of Table 16 show that except MPNN, the results obtained using the other three algorithms are feasible.

This design problem has also been solved using the five recently proposed optimization techniques, such as CA, GOA, SSO, SCA, and SGA. These algorithms have been run for several times and the best obtained solutions are reported in Table 17 with a precision level up to 4 digits after the decimal point. From the results, it is observed that all the results are feasible and the amount of constraint violations (CV) are found to be zero. Moreover, the proposed RGA-DX has yielded the best optimized results for this problem of interest.

The directional information guides the search process to concentrate on the most potential areas of the variable space, where the chance of getting good solutions is more. However, the search process can be directed randomly also with the use

Table 15 Comparison of optimization results for the cantilever beam design problem

Variables	RGA-DX	SDGAM _{INLS} [60]	MPNN [60]	SUMT [60]
b_1	3.0471	3.0459	3.0606	2.17
h_1	60.9417	60.8969	61.2115	42.74
b_2	2.8094	2.8018	2.8161	2.27
h_2	56.1876	56.0168	56.3214	44.99
b_3	2.5238	2.5251	2.5216	2.82
h_3	50.4757	50.4643	50.429	50.47
b_4	2.2063	2.2252	2.2136	2.79
h_4	44.1250	44.4745	44.2759	55.42
b_5	1.7498	1.7678	1.7503	3
h_5	34.9951	34.8462	35.0141	59.77
$f(x)$	62,952.6883	63,044.17	63,240.67	65,678

Table 16 Comparison of constrained values for the cantilever beam design problem using four different methods

Constrained values	RGA-DX	SDGAM _{INLS} [60]	MPNN [60]	SUMT [60]
g_1	-0.6023	-0.5814	-0.0703	-0.465
g_2	-0.29801	-0.2205	-0.0422	-0.068
g_3	-0.00153	-0.0020	0.0025	-0.003
g_4	-5.73E-07	-0.1158	-0.0125	-0.0002
g_5	-4.86E-06	-0.00370	-0.0014	-0.0003
g_6	-2.8E-07	-0.00074	-0.0094	-0.002
g_7	-0.0003	-0.02241	-0.0006	-0.524
g_8	-0.0004	-0.02071	-0.0005	-0.442
g_9	-0.0003	-0.03804	-0.0030	-0.0007
g_{10}	-0.001	-0.03109	0.0039	-0.375
g_{11}	-0.0009	-0.51026	0.0081	-0.222

of lower directional probability (p_d). The calculated parameter val , the multiplying factor (α), and the children identification conditions promote the search process to focus over a wider range in the variable space, while the higher value of p_d helps the search process to constrict over the relatively smaller region. These two conflicting forces actually support to maintain a proper balance between the diversification and intensification powers of the RGA-DX, due to which the proposed method is able to show the better performance compared to others.

6 Conclusions

In this paper, a novel recombination operator, namely directional crossover (DX) has been proposed and implemented in a real-coded genetic algorithm (RGA). The working principle of the developed crossover operator is dominated by the directional information of the optimization problem to be solved. This obtained information helps the search process to be directed towards the most potential areas of the variable

space. However, to maintain a proper balance between the exploration and the exploitation abilities of an RGA, a parameter, called directional probability (p_d), has been introduced. The performance of an RGA with the proposed DX operator (RGA-DX) has been tested on a set of six well-known single-objective optimization functions with the varying number of variables and the results are compared to that yielded result using the RGAs with four other popular crossover operators. Moreover, comparisons are made among the obtained results of five RGAs on the said test functions with initial variables' boundaries, which do not contain the globally optimum solution. In addition, a performance comparison is done among the results obtained using RGAs without any mutation operation. In all the experiments, the RGA-DX has outperformed the other RGAs. Furthermore, several non-parametric statistical analyses, such as pairwise and multiple comparisons have been carried out on the obtained results. From the study, it is obvious that the RGA-DX has yielded significantly superior performance compared to the others. Moreover, the convergence analysis using the Page's trend test has revealed that the RGA-DX provides the fastest convergence rate among the five

Table 17 Comparison of results of RGA-DX and that of recently developed optimization algorithms for the cantilever beam design problem

Variables	RGA-DX	CA	GOA	SSO	SCA	SGA
b_1	3.0471	3.6522	3.9108	3.0105	3.4054	3.0504
h_1	60.9417	54.2181	56.1624	60.1166	65.0000	61.0076
b_2	2.8094	2.8268	4.1213	2.7962	3.7089	2.8056
h_2	56.1876	55.3007	48.2459	55.6789	52.2143	56.1091
b_3	2.5238	2.6474	2.8667	2.5453	2.6129	2.5242
h_3	50.4757	52.9474	49.2895	50.5066	51.2293	50.4787
b_4	2.2063	2.2644	2.5772	2.2979	2.7031	2.2065
h_4	44.1250	45.1759	40.8266	45.7166	42.7503	44.1283
b_5	1.7498	2.0275	4.4926	1.8095	2.4297	1.7498
h_5	34.9951	40.5027	30.0000	36.1427	31.8199	34.9952
CV	0.0	0.0	0.0	0.0	0.0	0.0
$f(x)$	62,952.6883	67,892.7843	79,977.0265	63,167.0697	74,173.6802	62,953.9303

RGAs and it is followed by the other algorithms, such as RGA-RDD, RGA-RX, RG-SBX, and RGA-LX. This fact of having the better convergence rate compared to others has been again established through the time study, where the RGA-DX has taken the minimum CPU time and number of function evaluations to reach a particular accuracy of the objective function value for all six test functions. Moreover, a performance comparison of the proposed RGA-DX with five other recently developed optimization algorithms for all the test functions proves its supremacy over the others. Finally, the proposed RGA-DX has been implemented to the realm of engineering applications, where a constrained cantilever beam design problem has been solved using the RGA-DX and the results are compared with that of the obtained ones using other approaches. In this case, the proposed RGA-DX is able to find the better design of the cantilever beam compared to the others.

The use of the directional information helps the algorithm to search in more potential regions of the variable space, while the calculated parameter val , the parameter multiplying factor (α), and the child identification conditions promote the search process to spread over the undiscovered regions of the variable space. These are the reasons due to which a proper balance between the diversification and intensification powers of the proposed algorithm is maintained and consequently, the RGA-DX is able to produce superior results compared to others. In the future, the RGA-DX will be applied to solve other optimization functions and real-world problems to establish its supremacy in the broader sense.

References

- Gong X, Plets D, Tanghe E, De Pessemer T, Martens L, Joseph W (2018) An efficient genetic algorithm for large-scale planning of dense and robust industrial wireless networks. *Expert Syst Appl* 96:311–329
- Bermejo E, Campomanes-Álvarez C, Valsecchi A, Ibáñez O, Damas S, Cordon O (2017) Genetic algorithms for skull-face overlay including mandible articulation. *Inf Sci* 420:200–217. <https://doi.org/10.1016/j.ins.2017.08.029>
- Liao C-L, Lee S-J, Chiou Y-S, Lee C-R, Lee C-H (2018) Power consumption minimization by distributive particle swarm optimization for luminance control and its parallel implementations. *Expert Syst Appl* 96:479–491
- Fernández JR, López-Campos JA, Segade A, Vilán JA (2018) A genetic algorithm for the characterization of hyperelastic materials. *Appl Math Comput* 329:239–250. <https://doi.org/10.1016/j.amc.2018.02.008>
- Morra L, Coccia N, Cerquitelli T (2018) Optimization of computer aided detection systems: an evolutionary approach. *Expert Syst Appl* 100:145–156
- Gao H, Pun C-M, Kwong S (2016) An efficient image segmentation method based on a hybrid particle swarm algorithm with learning strategy. *Inf Sci* 369:500–521. <https://doi.org/10.1016/j.ins.2016.07.017>
- Wang JL, Lin YH, Lin MD (2015) Application of heuristic algorithms on groundwater pumping source identification problems. In: *International Conference on Industrial Engineering and Engineering Management (IEEM)*, 6–9 Dec 2015, pp 858–862. <https://doi.org/10.1109/IEEM.2015.7385770>
- Nazari-Heris M, Mohammadi-Ivatloo B (2015) Application of heuristic algorithms to optimal PMU placement in electric power systems: an updated review. *Renew Sust Energ Rev* 50:214–228. <https://doi.org/10.1016/j.rser.2015.04.152>
- Niu M, Wan C, Xu Z (2014) A review on applications of heuristic optimization algorithms for optimal power flow in modern power systems. *J Mod Power Syst Clean Energy* 2(4):289–297. <https://doi.org/10.1007/s40565-014-0089-4>
- Ghaheer A, Shoar S, Naderan M, Hoseini SS (2015) The applications of genetic algorithms in medicine. *Oman Med J* 30(6):406–416. <https://doi.org/10.5001/omj.2015.82>
- Reina DG, Ruiz P, Ciobanu R, Toral SL, Dorronsoro B, Dobre C (2016) A survey on the application of evolutionary algorithms for Mobile multihop Ad Hoc network optimization problems. *Int J Distrib Sens Netw* 12(2):2082496. <https://doi.org/10.1155/2016/2082496>
- Cordon O, Herrera-Viedma E, López-Pujalte C, Luque M, Zarco C (2003) A review on the application of evolutionary computation to information retrieval. *Int J Approx Reason* 34(2):241–264. <https://doi.org/10.1016/j.ijar.2003.07.010>
- Steinbuch R (2010) Successful application of evolutionary algorithms in engineering design. *J Bionic Eng* 7:S199–S211. [https://doi.org/10.1016/S1672-6529\(09\)60236-5](https://doi.org/10.1016/S1672-6529(09)60236-5)
- Ma R-J, Yu N-Y, Hu J-Y (2013) Application of particle swarm optimization algorithm in the heating system planning problem. *Sci World J* 2013:11. <https://doi.org/10.1155/2013/718345>
- Anis Diyana R, Nur Sabrina A, Hadzli H, Noor Ezzan A, Suhaimi S, Rohaiza B (2018) Application of particle swarm optimization algorithm for optimizing ANN model in recognizing ripeness of citrus. *IOP Conference Series: Materials Science and Engineering* 340(1):012015
- Assareh E, Behrang MA, Assari MR, Ghanbarzadeh A (2010) Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran. *Energy* 35(12):5223–5229. <https://doi.org/10.1016/j.energy.2010.07.043>
- Cao H, Qian X, Zhou Y (2018) Large-scale structural optimization using metaheuristic algorithms with elitism and a filter strategy. *Struct Multidiscip Optim* 57(2):799–814. <https://doi.org/10.1007/s00158-017-1784-3>
- Schutte JF, Koh B, Reinbolt JA, Haftka RT, George AD, Fregly BJ (2005) Evaluation of a particle swarm algorithm for biomechanical optimization. *J Biomech Eng* 127(3):465–474
- Das AK, Pratihar DK (2018) A novel restart strategy for solving complex multi-modal optimization problems using real-coded genetic algorithm. In: Abraham A, Muhuri P, Muda A, Gandhi N (eds) *Intelligent systems design and applications*. ISDA 2017. *Advances in Intelligent Systems and Computing*, vol 736. Springer, Cham
- Das AK, Pratihar DK (2018) Performance improvement of a genetic algorithm using a novel restart strategy with elitism principle. *International Journal of Hybrid Intelligent Systems (Pre-press)*:1–15. <https://doi.org/10.3233/HIS-180257>
- Kogiso N, Watson LT, Gürdal Z, Haftka RT (1994) Genetic algorithms with local improvement for composite laminate design. *Structural Optimization* 7(4):207–218. <https://doi.org/10.1007/bf01743714>
- Kogiso N, Watson LT, Gürdal Z, Haftka RT, Nagendra S (1994) Design of composite laminates by a genetic algorithm with memory. *Mech Compos Mater Struct* 1(1):95–117. <https://doi.org/10.1080/10759419408945823>
- Soremekun G, Gürdal Z, Haftka RT, Watson LT (2001) Composite laminate design optimization by genetic algorithm with generalized

- elitist selection. *Comput Struct* 79(2):131–143. [https://doi.org/10.1016/S0045-7949\(00\)00125-5](https://doi.org/10.1016/S0045-7949(00)00125-5)
24. Mestria M (2018) New hybrid heuristic algorithm for the clustered traveling salesman problem. *Comput Ind Eng* 116:1–12. <https://doi.org/10.1016/j.cie.2017.12.018>
 25. Nama S, Saha AK (2018) A new hybrid differential evolution algorithm with self-adaptation for function optimization. *Appl Intell* 48(7):1657–1671. <https://doi.org/10.1007/s10489-017-1016-y>
 26. Singh A, Banda J (2017) Hybrid artificial bee colony algorithm based approaches for two ring loading problems. *Appl Intell* 47(4):1157–1168. <https://doi.org/10.1007/s10489-017-0950-z>
 27. Srivastava S, Sahana SK (2017) Nested hybrid evolutionary model for traffic signal optimization. *Appl Intell* 46(1):113–123. <https://doi.org/10.1007/s10489-016-0827-6>
 28. Canayaz M, Karci A (2016) Cricket behaviour-based evolutionary computation technique in solving engineering optimization problems. *Appl Intell* 44(2):362–376. <https://doi.org/10.1007/s10489-015-0706-6>
 29. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. *Adv Eng Softw* 105:30–47. <https://doi.org/10.1016/j.advengsoft.2017.01.004>
 30. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. *Knowl-Based Syst* 96:120–133. <https://doi.org/10.1016/j.knsys.2015.12.022>
 31. Cuevas E, Cienfuegos M, Zaldívar D, Pérez-Cisneros M (2013) A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Syst Appl* 40(16):6374–6384. <https://doi.org/10.1016/j.eswa.2013.05.041>
 32. Gonçalves MS, Lopez RH, Miguel LFF (2015) Search group algorithm: a new metaheuristic method for the optimization of truss structures. *Comput Struct* 153:165–184. <https://doi.org/10.1016/j.compstruc.2015.03.003>
 33. Verma R, Lakshminarayanan PA (2006) A case study on the application of a genetic algorithm for optimization of engine parameters. *Proc IMechE, Part D: J Automobile Engineering* 220(4):471–479. <https://doi.org/10.1243/09544070D09204>
 34. Wu H, Hsiao W, Lin C, Cheng T (2011) Application of genetic algorithm to the development of artificial intelligence module system. In: 2nd international conference on intelligent control and information processing, 25–28 July 2011. pp 290–294. <https://doi.org/10.1109/ICICIP.2011.6008251>
 35. Canyurt OE, Öztürk HK (2006) Three different applications of genetic algorithm (GA) search techniques on oil demand estimation. *Energy Convers Manag* 47(18):3138–3148. <https://doi.org/10.1016/j.enconman.2006.03.009>
 36. Barros GAB, Carvalho LFBS, Silva VRM, Lopes RVV (2011) An application of genetic algorithm to the game of checkers. In: Brazilian symposium on games and digital entertainment, 7–9 Nov. 2011. pp 63–69. <https://doi.org/10.1109/SBGAMES.2011.14>
 37. Holland JH (1992) *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence.* MIT Press, Cambridge
 38. Goldberg DE (1989) *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Longman Publishing Co., Boston
 39. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) *Section 22.3, gray codes, Numerical recipes: the art of scientific computing, 3rd edn.* Cambridge University Press, New York
 40. MacKay DJ, Mac Kay DJ (2003) *Information theory, inference and learning algorithms.* Cambridge university press, Cambridge
 41. Das AK, Pratihar DK (2017) A direction-based exponential crossover operator for real-coded genetic algorithm. Paper presented at the seventh international conference on theoretical, applied, computational and experimental mechanics, IIT Kharagpur, India
 42. Herrera F, Lozano M (2000) Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. *Appl Intell* 13(3):187–204. <https://doi.org/10.1023/a:1026531008287>
 43. Herrera F, Lozano M, Verdegay JL (1998) Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. *Artif Intell Rev* 12(4):265–319
 44. Jomikow C, Michalewicz Z (1991) An experimental comparison of binary and floating point representations in genetic algorithm. In: Proceedings of the fourth international conference on genetic algorithms, pp 31–36
 45. Chuang Y-C, Chen C-T, Hwang C (2015) A real-coded genetic algorithm with a direction-based crossover operator. *Inf Sci* 305:320–348. <https://doi.org/10.1016/j.ins.2015.01.026>
 46. Wright AH (1991) Genetic algorithms for real parameter optimization. In: Rawlins GJE (ed) *Foundations of genetic algorithms, vol 1.* Elsevier, pp 205–218. <https://doi.org/10.1016/B978-0-08-050684-5.50016-1>
 47. Radcliffe NJ (1991) Equivalence class analysis of genetic algorithms. *Complex Syst* 5(2):183–205
 48. Michalewicz Z (1996) *Genetic algorithms + data structures = evolution programs, 3rd edn.* Springer-Verlag, New York
 49. Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. In: Whitley LD (ed) *Foundations of genetic algorithms, vol 2.* Elsevier, pp 187–202. <https://doi.org/10.1016/B978-0-08-094832-4.50018-0>
 50. Voigt H-M, Mühlenbein H, Cvetkovic D (1995) Fuzzy recombination for the breeder genetic algorithm. In: Proceedings of the 6th international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., pp 104–113
 51. Deb K, Agrawal RB (1994) Simulated binary crossover for continuous search space. *Complex Syst* 9(3):1–15
 52. Ono I, Kita H, Kobayashi S (2003) A real-coded genetic algorithm using the unimodal Normal distribution crossover. In: Ghosh A, Tsutsui S (eds) *Advances in evolutionary computing: theory and applications.* Springer Berlin Heidelberg, Berlin, pp 213–237. https://doi.org/10.1007/978-3-642-18965-4_8
 53. 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 evol comput - volume 1, Orlando, Florida, 1999. Morgan Kaufmann Publishers Inc., San Mateo, CA, pp 496–503
 54. Kita H, Ono I, Kobayashi S (1999) Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In: Proceedings of the 1999 congress on evolutionary computation, pp 1588–1595. <https://doi.org/10.1109/CEC.1999.782672>
 55. Herrera F, Lozano M (1996) Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing* 8:95–125
 56. Herrera F, Lozano M, Verdegay JL (1996) Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms. *Int J Intell Syst* 11(12):1013–1040
 57. 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 evol comput - volume 1, Orlando, Florida. Morgan Kaufmann Publishers Inc., pp 657–664
 58. Deb K, Anand A, Joshi D (2002) A computationally efficient evolutionary algorithm for real-parameter optimization. *Evol Comput* 10(4):371–395. <https://doi.org/10.1162/106365602760972767>
 59. Deep K, Thakur M (2007) A new crossover operator for real coded genetic algorithms. *Appl Math Comput* 188(1):895–911. <https://doi.org/10.1016/j.amc.2006.10.047>

60. Kuo H-C, Lin C-H (2013) A directed genetic algorithm for global optimization. *Appl Math Comput* 219(14):7348–7364. <https://doi.org/10.1016/j.amc.2012.12.046>
61. Lim SM, Sulaiman MN, Sultan ABM, Mustapha N, Tejo BA (2014) A new real-coded genetic algorithm crossover: Rayleigh crossover. *Journal of Theoretical & Applied Information Technology* 62(1):262–268
62. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82
63. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. *Adv Eng Softw* 69:46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
64. Deb K, Goyal M (1996) A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics* 26:30–45
65. Deb K (2000) An efficient constraint handling method for genetic algorithms. *Comput Methods Appl Mech Eng* 186(2):311–338. [https://doi.org/10.1016/S0045-7825\(99\)00389-8](https://doi.org/10.1016/S0045-7825(99)00389-8)
66. Van Den Bergh F, Engelbrecht AP (2006) A study of particle swarm optimization particle trajectories. *Inf Sci* 176(8):937–971
67. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol Comput* 1(1):3–18
68. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf Sci* 180(10):2044–2064
69. Derrac J, García S, Hui S, Suganthan PN, Herrera F (2014) Analyzing convergence performance of evolutionary algorithms: a statistical approach. *Inf Sci* 289:41–58. <https://doi.org/10.1016/j.ins.2014.06.009>
70. Deb K, H-g B (2001) Self-adaptive genetic algorithms with simulated binary crossover. *Evol Comput* 9(2):197–221. <https://doi.org/10.1162/106365601750190406>



Amit Kumar Das received his B.E. and M. Tech. from Indian Institute of Engineering Science and Technology, Shibpur, West Bengal, India (formerly Bengal Engineering & Science University, Shibpur) and IIT Kharagpur, India, in 2008 and 2016, respectively. He is currently pursuing his Ph.D. at IIT Kharagpur, West Bengal, India.



Dilip Kumar Pratihar received his BE (Hons.) and M. Tech. from REC (NIT) Durgapur, India, in 1988 and 1994, respectively. He obtained his Ph.D. from IIT Kanpur, India in 2000. He received University Gold Medal, A.M. Das Memorial Medal, Institution of Engineers (I) Medal, and others. He completed his post-doctoral studies in Japan and then in Germany under the Alexander von Humboldt Fellowship Programme. He is working as a Professor of IIT

Kharagpur, India. His research areas include robotics, soft computing and manufacturing science. He has published more than 240 papers, mostly in various international journals. He has written a textbook on “Soft Computing”, co-authored another textbook on “Analytical Engineering Mechanics”, edited a book on “Intelligent and Autonomous Systems”, co-authored reference books on “Modeling and Analysis of Six-legged Robots” and “Modeling and Simulations of Robotic Systems Using Soft Computing”. Recently, he has published another textbook named “Fundamentals of Robotics”. He has guided 18 Ph.D.s. He is in editorial board of 14 International Journals. He has been elected as FIE, SMIEEE, MASME.