Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 4243853, 17 pages
https://doi.org/10.1155/2019/4243853

Research Article

An Improved Real-Coded Genetic Algorithm Using the Heuristical Normal Distribution and Direction-Based Crossover

Jiquan Wang,[1] **Mingxin Zhang**,[1] **Okan K. Ersoy**,[2] **Kexin Sun**,[1] **and Yusheng Bi**[1]

[1]*College of Engineering, Northeast Agricultural University, Harbin, Heilongjiang 150030, China*
[2]*School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285, USA*

Correspondence should be addressed to Jiquan Wang; wang-jiquan@163.com

Received 15 July 2019; Revised 17 September 2019; Accepted 20 September 2019; Published 14 November 2019

Academic Editor: Paolo Gastaldo

Copyright © 2019 Jiquan Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A multi-offspring improved real-coded genetic algorithm (MOIRCGA) using the heuristical normal distribution and direction-based crossover (HNDDBX) is proposed to solve constrained optimization problems. Firstly, a HNDDBX operator is proposed. It guarantees the cross-generated offsprings are located near the better individuals in the population. In this way, the HNDDBX operator ensures that there is a great chance of generating better offsprings. Secondly, as iterations increase, the same individuals are likely to appear in the population. Therefore, it is possible that the two parents of participation crossover are the same. Under these circumstances, the crossover operation does not generate new individuals, and therefore does not work. To avoid this problem, the substitution operation is added after the crossover so that there is no duplication of the same individuals in the population. This improves the computational efficiency of MOIRCGA by leading it to quickly converge to the global optimal solution. Finally, aiming at the shortcoming of a single mutation operator which cannot simultaneously take into account local search and global search, a Combinational Mutation method is proposed with both local search and global search. The experimental results with sixteen examples show that the multi-offspring improved real-coded genetic algorithm (MOIRCGA) has fast convergence speed. As an example, the optimization model of the cantilevered beam structure is formulated, and the proposed MOIRCGA is compared to the RCGA in optimizing the parameters of the cantilevered beam structure. The optimization results show that the function value obtained with the proposed MOIRCGA is superior to that of RCGA.

1. Introduction

The genetic algorithm (GA) was proposed by Professor Holland and his students at the University of Michigan at the end of the 1960s and in the early 1970s [1–4]. In 1975, De Hong first proposed the evolutionary strategy (ES) of elitism preservation in his doctoral thesis. Later, others further studied it and proposed several elitism preservations and ES of replacing copying with selection [5–8]. At present, the GA is basically calculated according to these evolutionary strategies. In 1989, Goldberg [9] made a comprehensive and systematic summary and discussion of the GA and laid the foundation of the modern GA.

Initial GA uses binary coding. Because it can only map to discrete values in the search space and has Hamming distance, the accuracy of solution is not high [10]. In addition, binary coding needs to be encoded and decoded frequently, thus increasing the calculation time, with potential conversion error. Since the accuracy of the solution is controlled by the encoding length, for high accuracy, binary coding may need too long codes, resulting in excessive computing and memory space as well as reduced computational speed. In 1989, Lucasius et al. first proposed real-coded genetic algorithms (RCGAs) [11, 12]. RCGA has many attractive properties such as high precision, no coding and decoding required, effective large space search, simple computing, fast convergence, and not easy to fall into a local extreme value [13, 14].

GA has been widely applied in areas such as function optimization, automatic control, machine learning, combinatorial optimization, production scheduling problems, image processing, self-adaptation control, planning and design, industrial engineering, intelligent manufacturing systems, bioengineering, system engineering, artificial

intelligence, intelligent machine system, artificial life, text information filtering, and cooperative multi-objective evaluation [15–22]. It is especially suitable to deal with complex and nonlinear continuous optimization problems that cannot be solved by traditional search methods [6].

During the last few decades, the RCGA has attracted a lot of attention because of its unique and excellent performance. Many scholars have made in-depth research on RCGA and achieved excellent research results [23–31]. Because crossover operators and mutation operators have a great influence on the performance of GA, many scholars have focused their attention on the improvement of crossover operators and mutation operators. In 1992, Wright [32] proposed a heuristic crossover (HX) operator, which is used to solve constrained and unconstrained optimization problems and achieved good results. The two offsprings generated by the HX operator are located on the straight line connecting the two parents, and are in the vicinity of the parent individuals with better fitness value. In 1993, Eshelman et al. [23] used the concept of interval schemata to develop a blend crossover operator (BLX-α). When the difference between the parents is small, the offsprings are similar to the parents, but if the difference between the parents is large, the generation of the offspring is similar to a random search. The parameter α is used to control the search domain, and the optimal value of α is suggested to be 0.5; BLX-α generates the offspring solution at the center from both selected parents. In 1995, Deb et al. [33] proposed a simulated binary crossover (SBX) which simulates the single-point binary crossover and generates two offsprings from two selected parents. The two offsprings generated by the SBX operator are located on the straight line connecting the two parents and are in the vicinity of the two parents. The distribution index η in the SBX operator controls the distance between two offsprings and two parents. If the value of η is large, the probability that the generated offsprings are closer to the two parents is greater; if the value of η is relatively small, the probability that the generated offsprings are farther away from the two parents is greater. The disadvantage of SBX is that it cannot adaptively control the size of the parameter η value, and therefore cannot adaptively control the distance between two offsprings generated by the crossover and the two parents. In 1997, Ono et al. [34] proposed the unimodal normal distribution crossover (UNDX) operator which used an ellipsoidal probability distribution to produce two or more offsprings from three selected parents. In this scheme, the offsprings located near the center of the first two parents are produced with a higher probability whereas those neighboring the parents are produced with a lower probability. In 2002, Deb et al. [35] developed a parent-centric crossover (PCX) operator. PCX is a self-adaptive multi-parent crossover which uses a large probability rather than the center of selected parents to generate a new solution near each parent. One parent is selected for each generated offspring and a difference vector is calculated between this parent and the N chosen parents. Besides, the PCX has difficulties to solve multimodal problems. The comparison results show that RCGA based on PCX outperforms other comparative crossover schemes in finding the global optimum for a number of test problems. In 2007, Deep et al. [36] proposed a Laplace crossover (LX) in which Laplace distribution is used as the density function to decide the location of the offspring. The two offsprings generated by the LX operator are symmetrical relative to their parental position, and the two offsprings are not necessarily located near the better among the two parents. In 2016, Chuang et al. [27] proposed a direction-based crossover (DBX) which is able to explore $2^n - 1$ possible search directions. As a result, a high probability to produce better offspring individuals is assured. However, the search directions of DBX are limited. Although it may generate a crossover direction capable of guiding the chromosomes to move towards the optimal solution; this is not highly probable. Meanwhile, when the dimensionalities of the variables are small, the null vector solution is likely to be generated. Currently, the search direction is randomly generated, and the search direction is not instructive. In 2018, Wang et al. [37] developed a heuristic normal distribution crossover (HNDX) operator. It can guarantee the cross-generated offspring to locate closer to the better one among the two parents and the search direction to be very close to the optimal search direction or to be consistent with the optimal searching direction. However, HNDX does not consider whether there are better individuals in the population than the two parents of participation crossover, and if so, whether it should be considered in the crossover operator.

Similarly, several other mutation schemes are reported in the literature. In 1996, Michalewicz [38] proposed a nonuniform mutation (NUM) operator. At the beginning of the iterations, NUM's global search ability is strong; in the latter part of the iterations, NUM's local search ability is strong. The disadvantage of the NUM operator is that it is difficult to choose a maximum number of iterations that applies to all problems. In 1995, Hinterding [39] proposed a Gaussian mutation operator in which the global and local search capabilities of GA are changed by adjusting the variance of the Gaussian mutation operator. The disadvantage of Gaussian mutation is that the local search ability is stronger than the global search ability, and the algorithm is easy to fall into local optimum. Therefore, Gaussian mutation is not suitable for solving multimodal optimization problems. In 1996, Wang et al. [40] proposed a mutation operator which mutated towards the gradient direction of the objective function. However, when the objective function is not differentiable, the mutation operator cannot be performed. In 2014, Deb et al. [41] proposed a polynomial mutation (PM) operator, which is widely used in RCGA. PM operators have strong random search ability and are not easy to fall into local optimum, but the convergence speed of the algorithm is slower. In 2016, Chuang et al. [27] proposed a dynamic random mutation (DRM) operator. Here, the calculation formula of the step size contradicts its interpretation. In addition, the mutation operator needed the maximum number of iterations in advance, which is difficult to estimate.

In the improved RCGA mentioned above, two parents generate two offsprings by crossover. In the process of biological evolution, a pair of parents usually breed more than

two offsprings. Such species not only survive well in the process of evolution, but also make the species evolve so as to get better species. In fact, there is no species in which the number of offsprings bred by a pair of parents is less than or equal to two. Even if such species exist, in the process of evolution, they will eventually become extinct due to diseases, lack of food, and other factors. In addition, when the number of offsprings within a population is greater than that of the parents, the population size will expand. If the survival space of the species is constant, the competition within the population will aggravate. In this case, excellent individuals with strong viability within the population are easy to survive, and individuals with lower viability are eliminated, so that the species gradually evolves to adapt to the living environment and get better species. Therefore, during the evolution of species in nature, the number of offsprings should be greater than that of the parents.

Inspired by the laws of biological evolution in nature, in 2015, Wang et al. [42] proposed a multi-offspring GA with single-point crossover and verified its effectiveness of single-point crossover multi-offspring GA by using test functions. In 2016, Wang et al. [43] developed multi-offspring GA. In this, the theoretical basis of multi-offspring GA was given. The schema theorem proved that multi-offspring GA is better than non-multi-offspring GA. This paper is only suitable for solving TSP. In 2016, Wang et al. [44] proposed a multi-offspring GA of two-point crossover and verified its validity using test functions. The multi-offspring GA given in the literature is mainly divided into two categories: one is the multi-offspring GA for solving the TSP; the other is the multi-offspring GA with binary coding.

In summary, the two offsprings generated by the LX operator are symmetric with respect to the parental position, and the two offsprings are not necessarily located near the better individuals of the two parents. The two offsprings generated by the SBX operator are located on the straight line connecting the two parents, and are in the vicinity of the two parent individuals. The two offsprings generated by the HX operator are located on the straight line connecting the two parents, and are in the vicinity of the parent individuals with better fitness value. The offspring individuals generated by PCX operators are not close to the center of the parental individuals, but close to the parental individuals. The two offsprings generated by the BLX-α operator locate at the center of the two parents of participation crossover. Since the DBX operator is only able to explore $2^n - 1$ possible search directions, the search directions of DBX are limited when the dimensionalities of the variables are small, and the null vector solution is likely to be generated. Currently, the search direction is randomly generated, and the search direction is not instructive. In order to overcome the shortcomings of the above crossover operator, this paper develops a heuristic normal distribution and direction-based crossover (HNDDBX) operator. HNDDBX not only considers the optimal individuals in the population, but also ensures that the offsprings generated after crossover are located near the better parent among the parents of participation crossover, not only near the better parent on the straight line connecting the two parents. In addition, a single mutation operator is difficult to take into account both global and local search abilities. For this reason, this paper develops a method of Combinational Mutation which includes three-mutation operators. Furthermore, because multi-offspring GA is superior to non-multi-offspring GA, the number of offsprings generated by the HNDDBX operator is more than those of the parents. Taking the above factors into consideration, this paper proposes a MOIRCGA, and it is compared with other GAs in the literature. The comparison results show that the performance of MOIRCGA is better than those of other GAs in the literature. Finally, MOIRCGA is applied to the optimization of the cantilever beam structure, and good results are obtained.

2. Penalty Function Method for Constrained Optimization Problems

The mathematic model of a constrained optimization problem can be generally expressed as follows:

$$\begin{cases} \min f(X), \\ \text{s.t.} \begin{cases} h_i(X) = 0, & i = 1, 2, \ldots, p, \\ g_j(X) \geq 0, & j = 1, 2, \ldots, q, \end{cases} \end{cases} \tag{1}$$

where n is the population size, $h_i(X) = 0$ is the i-th equality constraint, p is the number of equality constraints, $g_j(X) \geq 0$ is the j-th inequality constraint, q is the number of inequality constraints, and X_k is a D-dimensional vector $X_k = (x_{k1}, x_{k2}, \ldots, x_{kD})$.

Equation (1) can also be expressed as

$$\begin{cases} \min f(X),\ X = [X_1, X_2, \ldots, X_k, \ldots, X_n] \in R, \\ R = \left\{ X \mid h_i(X) = 0, i = 1, 2, \ldots, p; g_j(X) \geq 0, j = 1, 2, \ldots, q \right\}. \end{cases} \tag{2}$$

Stating X^* is the optimal solution to the constrained optimization problems means that $\forall X \in R$: $f(X^*) \leq f(X)$. In addition, if $g_j(X^*) = 0$, the constraint is referred to as the active constraint. Under this concept, all the equation constraints $h_i(X) = 0$ ($i = 1, 2, \ldots, p$) are active at X^*.

The penalty function method converts a constrained optimization problem to an unconstrained optimization problem by using two penalty factors and defining the penalty function to be minimized as [45]:

$$P(X, M) = f(X) + M_1 \sum_{i=1}^{p} \left[h_i(X) \right]^2 + M_2 \sum_{j=1}^{q} \left[\min\left(0, g_j(X)\right) \right]^2, \tag{3}$$

where M_1 and M_2 are the penalty factors, generally chosen as positive constants that are big enough; the second and third terms on the right are the penalty terms, and $P(X, M)$ is the penalty function.

In equation (3), when $X \in R$, there should be no penalty to the feasible points, thus $P(X, M) = f(X)$; when $X \notin R$, for the nonfeasible points, M_1 and M_2 should be very big. Moreover, when point X gets farther away from the feasible region, the penalty should be larger.

The minimum value of equation (3), that is,

$$\min P(X, M), \tag{4}$$

is equivalent to the minimum value of equation (1).

3. Multi-offspring Improved Real-Coded Genetic Algorithm (MOIRCGA)

In the multi-offspring GA, the number of offsprings is an integer multiple of the number of parents. Other GAs will be called conventional GAs.

Letting n_0 and n_1 be the number of parents and offsprings, respectively, we get

$$n_1 = kn_0, \quad k \in \{2, 3, 4, \ldots\}. \tag{5}$$

In the paper, the number of offsprings generated by crossover will be chosen as $2n$. In comparing multi-offspring GA with conventional GA, the competition among individuals in the population in multi-offspring GA is more intensifier. Most of the surviving individuals in the population are excellent. This causes multi-offspring GA to have faster convergence speed than conventional GA.

3.1. Mathematical Symbols of MOIRCGA. We define the following symbols: $X^0 = (X_1^0, X_2^0, \ldots, X_i^0, \ldots, X_n^0)$ is the initial population, where $X_i^0 = (X_{i1}^0, X_{i2}^0, \ldots, X_{iD}^0)$, $i = 1, 2, \ldots, n$; D and n are the dimensions of the variables and the population size, respectively; $a = [a_1, a_2, \ldots, a_D]$ and $b = [b_1, b_2, \ldots, b_D]$ are lower and upper bounds of the variables X_i^0. In the t-th iteration, $X_i(t)$ is also referred to as a chromosome with $x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t)$ known as genes. We also define P_c and P_m as the crossover mutation probabilities, respectively, and m is the number of preserved elitism individuals.

3.2. Initialization of Population. Before the population is initialized, the upper and lower bounds of the variables are determined. Hereafter, the variables are randomly generated uniformly within the interval $[a, b]$. Therefore, the i-th individual X_i^0 in the initial population is initialized as

$$X_i^0 = a + (b - a) * \text{rand}(1, D), \quad i = 1, 2, \ldots, n, \tag{6}$$

where $(b - a). * \text{rand}(1, D)$ is the product of multiplying $(b - a)$ with the element at the same position of rand(1, D), and rand(1, D) represents D uniformly distributed random numbers in [0, 1].

3.3. ES of MOIRCGA. In the population evolution process of MOIRCGA, the population size is assumed to be constant. The ES of MOIRCGA is as follows:

(1) Generate the initial population of size n, and sort all individuals in descending order according to their objective function values.

(2) Select m elitist individuals with best ranking among n parents. Use the method of sorting grouping selection (SGS) to select the parents of participation crossover as detailed in Section 3.4.

(3) Let the crossover probability P_c equal 1 and n parents generate $2n$ offsprings by crossover as detailed in Section 3.5.

(4) Generate a new population consisting of $2n$ offsprings and m elitist individuals chosen in step 3.

(5) Sort the individuals of the new population in descending order according to their objective function values.

(6) Select m elitist individuals with the best ranking in the population.

(7) Using the mutation operator, modify the $2n$ offsprings generated by crossover. This is discussed in Section 3.6.

(8) Regenerate the new population consisting of the $2nP_m$ mutated offsprings, $2n(1 - P_m)$ unmutated offsprings, and m elitist individuals chosen in step 6.

(9) Sort the individuals of the regenerated new population in descending order according to their objective function values.

(10) If the iterative termination condition is met, output the optimal solution and the optimal value. Otherwise, go to the next step.

(11) Select n individuals with best ranking within the last population. Go to step 2 to start the next iteration.

The ES flow diagram for MOIRCGA is shown in Figure 1.

In the ES above, because the crossover probability is 1, the number of new individuals generated by crossover increases, and thus the possibility of generating excellent individuals increases. In addition, the number of new individuals generated by mutation is $2nP_m$, which is twice the number of new individuals generated by conventional GA mutation. Therefore, the possibility of obtaining excellent individuals and jumping out of local optimum is increased. This helps to improve the convergence speed of MOIRCGA.

3.4. Sorting Grouping Selection (SGS). We assume the following: (1) the maximum of the objective function is sought, (2) the population size n is an even number, and (3) the individuals in the population are sorted with respect to their objective values $P(X, M)$ in descending order. If the minimum of the objective function is sought, the objective function $P(X, M)$ can be changed to the maximum value by $\max P(X, M) = -\min[-P(X, M)]$. The population before sorting is $X(t) = (X_1(t), X_2(t), \ldots, (X_n(t))$, and after sorting in descending order becomes $X'(t) = (X_1'(t), X_2'(t), \ldots, (X_n'(t))$, where $P((X_1'(t), M) \geq P((X_2'(t), M) \geq \cdots \geq P((X_n'(t), M)$, t being the number of iterations.

The individuals in the population are divided into two groups according to their objective function values. Group 1 includes the better $n/2$ individuals, that is, $X_1'(t), X_2'(t), \ldots, X_{n/2}'(t)$, and Group 2 includes the other $n/2$ individuals, that is, $X_{n/2+1}'(t), X_{n/2+2}'(t), \ldots, X_n'(t)$. The 1st individual in Group 1 is matched with the 1st individual in Group 2, the 2nd

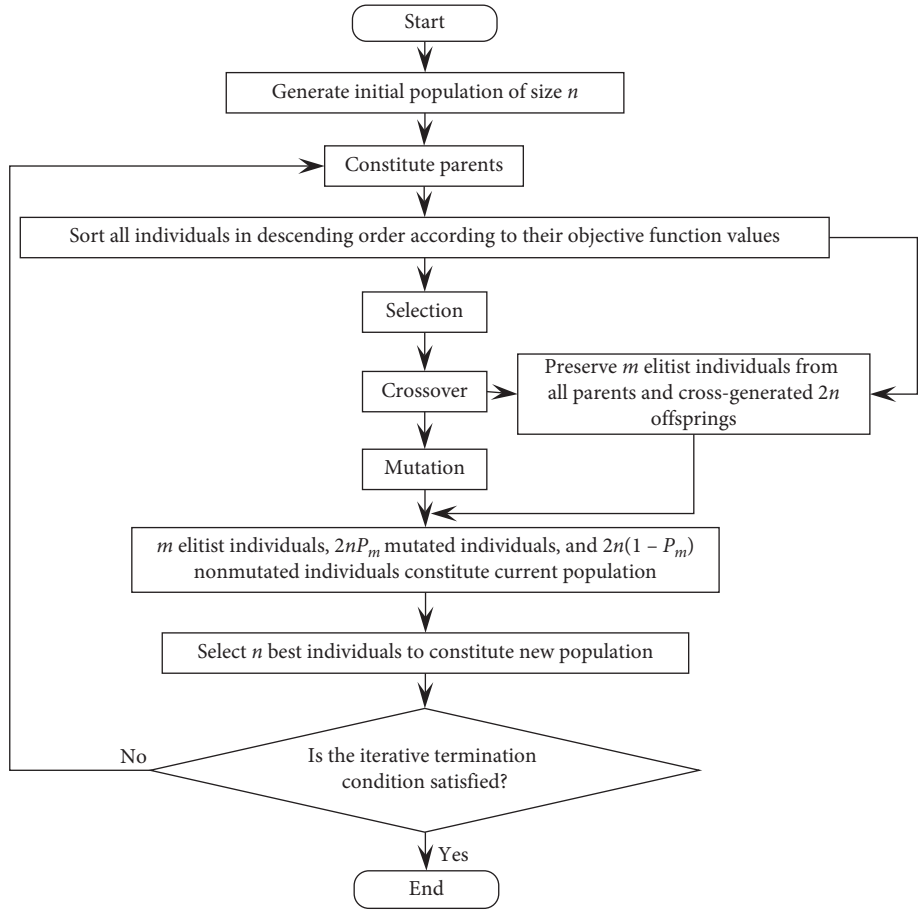


Figure 1: The flow diagram of MOIRCGA.

individual in Group 1 is matched with the 2nd individual in Group 2, and so on. In this way, we obtain $0.5n$ pairs of individuals of participation crossover. This selection method is called SGS. The advantage of the SGS method is that the selection operation is completed according to the objective function value of each individual, and it is not necessary to calculate the fitness value of each individual. This causes selection operation speed to be fast.

3.5. Heuristic Normal Distribution and Direction-Based Crossover (HNDDBX). If the optimization problem is to solve for the maximum value of the objective function, the optimal solution is likely to be near the individual with a bigger objective function value [46]. Hence, the better individuals in the population are more likely to approach the optimal solution. Inspired by this, this paper develops a heuristic normal distribution and direction-based crossover (HNDDBX) operator.

The key to HNDDBX is how to make the crossover generated offspring be in the vicinity of the better individuals in the population. Since a random number generated by the normal distribution has a higher probability of being located near its mean μ, if a better individual in the population is used as being the mean μ of the normal distribution, then it is guaranteed that its offspring according to normal distribution is in the vicinity of the better individual. Therefore, a normal distribution can be used to generate offsprings. The normal distribution is denoted by $N(\mu, \sigma^2)$, where μ is the mean, and σ^2 is the variance. The density function of the normal distribution is shown in Figure 2.

It can be seen from Figure 2 that the probability of a random number X generated by $N(\mu, \sigma^2)$ in the interval $(\mu - \sigma, \mu + \sigma)$ is 68.26%, the probability of X in the interval $(\mu - 2\sigma, \mu + 2\sigma)$ is 95.44%, and the probability of X in the interval $(\mu - 3\sigma, \mu + 3\sigma)$ is 99.72%. It can be seen that the probability that X falls outside the interval $(\mu - 3\sigma, \mu + 3\sigma)$ is less than 0.3%. Thus, the interval $(\mu - 3\sigma, \mu + 3\sigma)$ can be regarded as the actual possible interval of the random variable X, which is called the 3σ principle of the normal distribution. Based on the above analysis, as long as we control the size of σ value, we can basically guarantee that the random number X generated by $N(\mu, \sigma^2)$ is near the mean μ. If the mean μ of the normal distribution is equal to the better individual X within the parents of participation crossover, it can be ensured that the individuals generated by the normal distribution are located near the better individual X.

In order to explain the HNDDBX operator, we assume that the population size n is even. After sorting in descending order according to objective function values, the population is $X' = (X'_1, X'_2, \ldots, X'_n)$. At the t-th iteration, the optimal

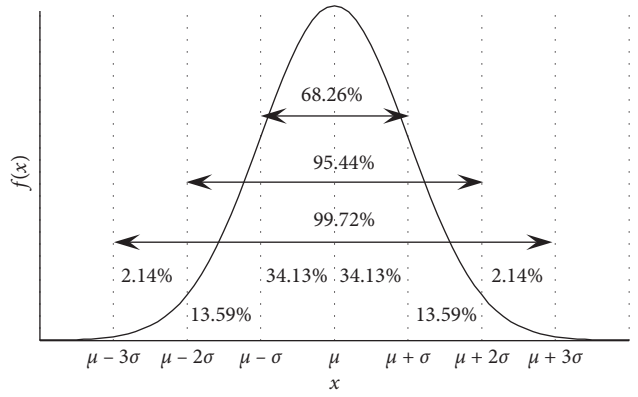


Figure 2: Density function of the normal distribution.

individual in the population is X_1'. This paper uses the SGS method to select individuals of participation crossover. Let the paired parents of participation crossover be X_i' ($i = 1, 2, \ldots, n/2$) and X_j' ($j = n/2+1, n/2+2, \ldots, n$). Assuming that X_i' is superior to X_j', the offsprings are generated by

$$\begin{cases} Y_i = N\left(\overline{X}, \varepsilon + \left[\dfrac{X_i' - X_j'}{12}\right]^2\right), \\ Y_j = N\left(X_1', \varepsilon + \left[\dfrac{X_1' - \overline{X}}{12}\right]^2\right), \\ Y_k = X_1' + R_1 * \left(X_i' - X_j'\right), \\ Y_l = \overline{X} + R_2 * \left(X_1' - \overline{X}\right), \\ \overline{X} = \dfrac{1}{3}\left[\overline{X}_1 + X_1' + X_i'\right], \\ \overline{X}_1 = \dfrac{X_1' + X_2' + \cdots + X_{0.5n}'}{0.5n}, \\ R_1 = (r_{11}, r_{12}, \ldots, r_{1D}), \\ R_2 = (r_{21}, r_{22}, \ldots, r_{2D}), \\ i = 1, 2, \ldots, \dfrac{n}{2};\ j = \dfrac{n}{2}+1,\ \dfrac{n}{2}+2, \ldots, n; \\ k = n+1, n+2, \ldots, \dfrac{3n}{2};\ l = \dfrac{3n}{2}+1, \dfrac{3n}{2}+2, \ldots, 2n, \end{cases} \tag{7}$$

where D is the dimension of the variables, Y_i ($i = 1, 2, \ldots, 2n$) are the offsprings generated by the crossovers, r_{11}, r_{12}, …, r_{1D} and r_{21}, r_{22}, …, r_{2D} are uniformly distributed random numbers in [0, 1], ε is a small positive number to avoid that the variance of the normal distribution is 0. $*$ means that the elements at the same position in two matrices or vectors are multiplied.

To illustrate the principle of HNDDBX, let the variable dimension D be 2. At the t-th iteration, let the two parent individuals of participation crossover be X_i' and X_j', respectively. If X_i' is superior to X_j', X_1' is the best individual in the population, and X^* is the optimal solution to the problem to be solved. The principle of HNDDBX is shown in Figure 3.

In Figure 3, $A\overline{X} = B\overline{X} = X_1'C = X_1'G = E\overline{X} = FX_1' = |X_i' - X_j'|$ and $\overrightarrow{D} = \overrightarrow{D_5} = \overrightarrow{D_6} = X_i' - X_j'$. Equation (7) shows that $\overline{X}_1$ is the center of the better $0.5 * n$ individuals in the population, X_1' is the best individual in the population, X_i' is the better one among the two parent individuals of participation crossover, and $\overline{X}$ is the center of $\overline{X}_1$, X_1', and X_i'. So, in most cases, $\overline{X}$ is superior to X_i'. Offsprings Y_i ($i = 1, 2, \ldots, n/2$) are randomly generated according to $N(\overline{X}, \varepsilon + ((X_i' - X_j')/12)^2)$, with its mean equal to the mean of $\overline{X}$ and its variance equal to $\varepsilon + ((X_i' - X_j')/12)^2$. In addition, it can be seen from Figure 3 that the random number generated according to $N(\overline{X}, \varepsilon + ((X_i' - X_j')/12)^2)$ has high probability of being in the vicinity of the mean $\overline{X}$ of the normal distribution. According to the above analysis, the two parent individuals X_i' and X_j' generate offsprings Y_i, which have a great possibility to be located within a circle with $\overline{X}$ as its center and $\overline{X} + 3(\varepsilon + ((X_i' - X_j')/12)^2)$ as its radius, as shown in Figure 3. When the dimension of the variables is greater than 2, Y_i has a great possibility to be located within a sphere with $\overline{X}$ as its center and $\overline{X} + 3(\varepsilon + ((X_i' - X_j')/12)^2)$ as its radius. Based on the above analysis, as long as we control the size of variance $\varepsilon + ((X_i' - X_j')/12)^2$, the HNDDBX operator can basically guarantee that the offsprings generated by crossover are near the mean $\overline{X}$. Similarly, the offspring individual Y_j ($j = n/2+1, n/2+2, \ldots, n$) generated by crossover according to equation (7) is near the optimal individual X_1' in the population. Because X_1' is the best individual in the population, the offsprings generated by crossover with the HNDDBX operator are expected to be better than those of the crossover operator in [31, 37].

It can be seen from Figure 3 that $X_i' - X_j'$ is the search direction for Y_k ($k = n+1, n+2, \ldots, 1.5n$). When $r_{12} = 0$, since r_{11} is a random number uniformly distributed in [0, 1], and the search direction is $\overrightarrow{D_3}$, Y_k is uniformly distributed on the line segment $X_1'G$. When $r_{11} = 0$, because r_{12} is a random number uniformly distributed in [0, 1] and the search direction is $\overrightarrow{D_4}$, Y_k is uniformly distributed on the line segment $X_1'C$. When $r_{11} = r_{12}$, since r_{11} and r_{12} are random numbers uniformly distributed in [0, 1] and the search direction is $\overrightarrow{D_6}$, Y_k is uniformly distributed on the line segment $X_1'F$. When r_{11} and r_{12} are random numbers uniformly distributed in [0, 1], the search directions are any directions between $\overrightarrow{D_3}$ and $\overrightarrow{D_4}$. At this point, there are countless search directions, and Y_k may be located at any point within $X_1'CFG$. In addition, because X_1' is better than X_i', the offsprings Y_k generated by the HNDDBX operator have a great possibility to be superior to those of the crossover operator in [31, 37]. Thus, Y_k may be very close to the optimal solution X^* of the problem to be solved. When $r_{22} = 0$, since r_{21} is a random number uniformly distributed in [0, 1] and the search direction is $\overrightarrow{D_1}$, Y_l is uniformly

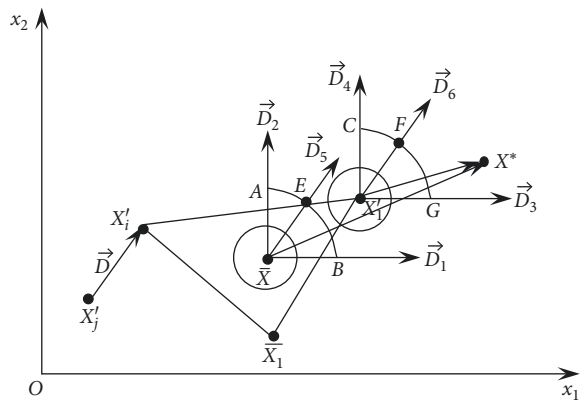


Figure 3: Schematic diagram of HNDDBX.

distributed on the line segment $\overline{X}B$. When $r_{21} = 0$, because r_{21} is a random number uniformly distributed in [0, 1] and the search direction is $\overrightarrow{D}_2$, Y_l is uniformly distributed on the line segment $\overline{X}A$. When r_{21} and r_{22} are random numbers uniformly distributed in [0, 1], the search directions are any directions between $\overrightarrow{D}_1$ and $\overrightarrow{D}_2$. At this point, there are countless search directions, and Y_l may be located at any point within $\overline{X}AEB$; Y_l may be very close to the optimal solution X^* of the problem to be solved.

In summary, the offsprings generated by the HNDDBX crossover operator are better than those of the crossover operator in [31, 37]. Therefore, the HNDDBX operator can significantly improve the convergence speed of MOIRCGA.

As an example, in terms of $2D$ search space, for the spatial positions of two parent individuals X_i' and X_j' participating in crossover, there are three possible cases: (a) both parents are in the infeasible region; (b) one parent is in the feasible region while the other is not; and (c) both parents are in the feasible region. Of these three cases, the HNDDBX operation in a two-dimensional space is shown in Figure 4.

In Figure 4, the schematic of case (a) shows that the HNDDBX operator can force two parents to move from the infeasible region to the feasible region. In the case of (b), the HNDDBX operator enables the individuals in the infeasible region to move into the feasible region, and the individuals that have been in the feasible region improve their objective function values by HNDDBX. When the individuals are already in the feasible region, as in the case of (c), HNDDBX searches near the better individual and the optimal individual moves towards the global optimal solution.

3.6. Substitution Operation. In the global optimization problem with many local optima, when the MOIRCGA finds a region with an extreme value (whether it is a local extremum or a global extremum), individuals in the population constantly move closer to the region, and they may be the same or similar individuals in the population. With the increase of the iterations, the same individuals in the population will gradually increase and may even make all the individuals in the population to be the same. If there are many such individuals in the population, it is likely that the two parents of participation crossover are the same, such that the offsprings Y_k generated by crossover, according to equation (7) are the same as the two parents of participation crossover. Hence, the crossover operation does not work, and GA is unable to converge to the global optimal solution. In order to avoid the above phenomenon, a substitution operation is added after crossover operation of GA. The method of substitution operation is as follows: the population size after crossover is $2n$; if there are two or more same individuals in the population, only one of those is retained, and the remaining individuals are removed. Let the population size be n_1 at this time. In order to maintain the population size after the crossover as $2n$, $2n - n_1$ individuals are randomly generated, and $2n - n_1$ same individuals are replaced by randomly generated $2n - n_1$ individuals. The following examples illustrate the substitution operation.

Suppose the population size is 10. After several iterations, the population is X. At this time, there are same individuals in the population. In order to improve the efficiency of crossover operation and avoid falling into local optimum, only one of the same individuals is retained; the population after removing the same individuals is X_1, which has four fewer individuals than X. In order to keep the size of the population unchanged, 4 individuals Y are randomly generated. The population X' after the substitution operation consists of X_1 and Y. X, X_1, and X' are shown in Table 1.

3.7. Combinational Mutation. With the mutation operators given in the existing literature, some local search abilities are strong [47, 48], such as Gauss mutation operator, and some global search abilities are strong [48], such as Cauchy mutation operator. For optimization problems with less number of extreme points, a mutation operator with stronger local search ability should be adopted. For optimization problems with more number of extreme points, if a mutation operator with stronger local search ability is adopted, it is easy to converge to the local optimum; if a mutation operator with strong global search ability is adopted, and the accuracy requirement of the optimal solution for the problem to be optimized is higher, the convergence speed of the algorithm slows down. With some of the mutation operators given in the literature, there is strong global search capability at the beginning of the iterations, and with the increase of the number of iterations, local search capabilities are enhanced. However, this requires the maximum number of iterations to be given, which is difficult to do in advance. Although this kind of mutation operator is theoretically feasible, it actually has poor performance [27]. In summary, a single mutation operator is difficult to take into account both global and local search abilities. For this reason, we developed a method of combinational mutation which includes three-mutation operators. They are as follows.

The first mutation operator is a mutation operator based on the Cauchy distribution called the Cauchy mutation operator. The Cauchy mutation operator is as follows:

$$\overline{X}_i' = \overline{X}_i + \overline{X}_i * \text{Cauchy}(0, 1), \tag{8}$$

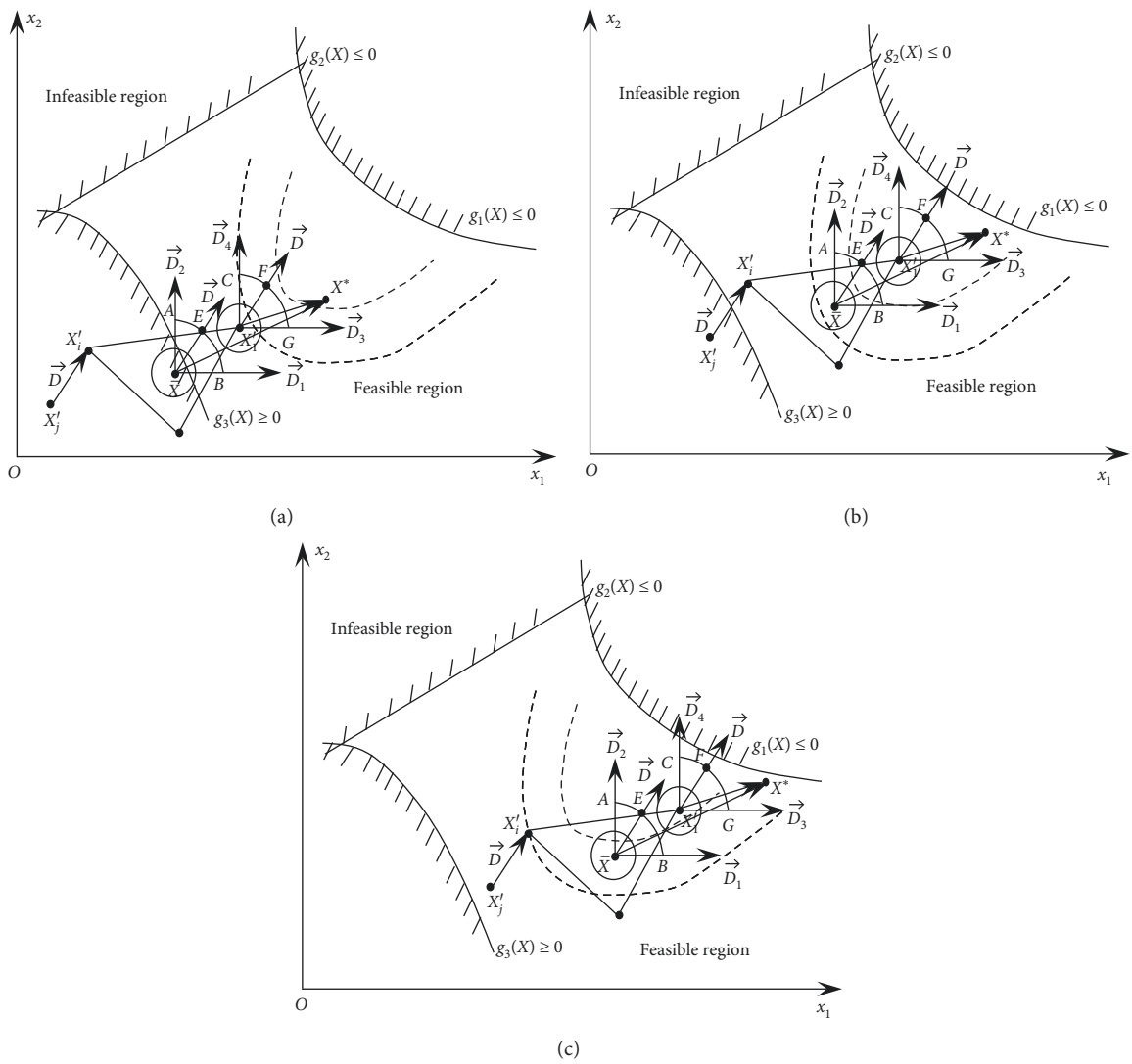


Figure 4: Schematic diagrams of the HNDDBX operations in a two-dimensional space under the following three cases: case (a): both parents are in the infeasible region; case (b): one parent is in the feasible region and the other one is in the infeasible region; case (c): both parents are in the feasible region.

Table 1: Substitution operation.

X	1	2	3	X_1	1	2	3	Y	2.3	4.6	−2.6	X'	1	2	3
	2	5	6		2	5	6		4	3.1	−2.5		2	5	6
	3	6	7		3	6	7		−3.1	4.8	3.6		3	6	7
	1	2	3		2.5	3.1	6.3		1.5	−2.4	4.5		2.5	3.1	6.3
	2.5	3.1	6.3		7.5	−6.5	8.6						7.5	−6.5	8.6
	7.5	−6.5	8.6		3.3	6.2	−4.8						3.3	6.2	−4.8
	3.3	6.2	−4.8										2.3	4.6	−2.6
	2	5	6										4	3.1	−2.5
	7.5	−6.5	8.6										−3.1	4.8	3.6
	2	5	6										1.5	−2.4	4.5

where Cauchy (0, 1) is the standard Cauchy distribution, $\overline{X}_i$ is the individual to be mutated, and $\overline{X}'_i$ is the individual after the mutation.

The second one is the mutation operator based on the normal distribution called the normal mutation. The normal mutation operator is as follows:

$$\overline{X}'_i = N\left(\mu, \delta^2\right), \tag{9}$$

where μ is the mean of the normal distribution, usually chosen as $\mu = \overline{X}_i$, and δ^2 is the variance of the normal distribution.

δ^2 of (9) is given by

$$\delta^2 = \left(\frac{X_{\text{best}} - \overline{X}_i}{12}\right)^2, \tag{10}$$

where X_{best} is the optimal individual in the population.

The third one is the mutation operator based on Lévy flight, called the Lévy mutation. It is as follows:

$$\overline{X}_i' = \overline{X}_i + \alpha * \text{Lévy}(\lambda), \tag{11}$$

where α is the step size scaling factor, generally chosen as $\alpha = 0.01$.

The calculation formula for the simulated Lévy flight path proposed by Mantegna is as follows [49]:

$$\text{Lévy}(\lambda) = \frac{\mu}{|v|^{1/\lambda}}, \tag{12}$$

where $0 < \lambda < 2$, usually chosen as $\lambda = 1.5$.

$$\begin{cases} \mu = N\left(0, \sigma_\mu^2\right), \\ v = N\left(0, \sigma_v^2\right), \end{cases}$$

$$\begin{cases} \sigma_\mu = \left\{\dfrac{\Gamma(1+\lambda)\sin(\pi\lambda/2)}{\Gamma[(1+\lambda)/2]\lambda 2^{(\lambda-1)/2}}\right\}^{1/\lambda}, \\ \sigma_v = 1, \end{cases} \tag{13}$$

where Γ is the Gamma function.

The two-dimensional space Lévy flight diagram is shown in Figure 5.

As seen in Figure 5, the characteristics of Lévy flight are as follows: (1) there are a lot of small steps, that is, most of the time, it gives a local search; (2) sometimes there is a large displacement, so that individuals in the population do not search only in one place, resulting in occasional global search capability.

The specific method of combinational mutation is as follows: in the t-th iterations, we divide the number of iterations by 3. When the remainder is 1, the first mutation operator is used; when the remainder is 2, the second mutation operator is used; when the remainder is 0, the third mutation operator is used.

Compared with the normal mutation operator, the generated offspring of Cauchy mutation operator has high probability of being far away from the individual to be mutated, so the global search ability of Cauchy mutation operator is strong. The normal mutation operator focuses on searching for a local region near the individual to be mutated, and the local search ability is stronger, but the ability to guide the individual to jump out of the local extremum is weak, which is not conducive to global convergence. The Lévy mutation operator performs local searches in most cases but occasionally performs global searches. Therefore, the advantage of combinational mutation is that both the local search ability and the global searching ability are taken into account.

3.8. Pseudocode of MOIRCGA. The pseudocode of MOIRCGA is shown in Algorithm 1.

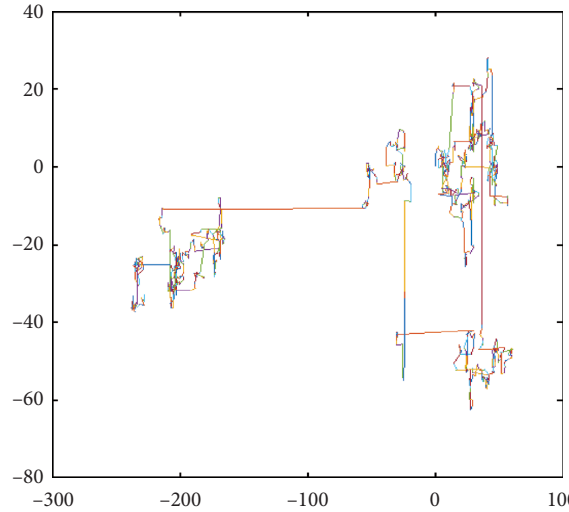


Figure 5: The two-dimensional space Lévy flight diagram.

4. Algorithmic Testing and Analysis

RCGA proposed in [27, 31, 37] is abbreviated as IRCGA-1, IRCGA-2, and IRCGA-3, respectively. The single-point crossover multi-offspring GA proposed in [42] is abbreviated as SPXMOGA, and hybrid GA in [50] is abbreviated as HGA.

In order to verify the validity and feasibility of MOIRCGA, it is compared with IRCGA-1, IRCGA-2, IRCGA-3, SPXMOGA, and HGA.

4.1. Iteration Termination Condition. The iteration termination condition for GA is to be defined as

$$\left|f_i - f_i^*\right| \le \varepsilon_i, \quad i = 1, 2, \ldots, p, \tag{14}$$

where f_i^* is the true global optimal value of the i-th test function, f_i is the optimal value of the i-th test function obtained by GA, and ε_i is the error for the i-th test function.

4.2. Test Functions. Sixteen well-known benchmark constrained functions were selected and are described in Appendix A [27, 31, 37].

4.3. Parameter Settings. In order to obtain a fair performance comparison, the parameters are set as follows.

(1) The computing precision of various test functions: $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = \varepsilon_6 = \varepsilon_7 = \varepsilon_8 = \varepsilon_9 = \varepsilon_{11} = \varepsilon_{12} = \varepsilon_{13} = \varepsilon_{14} = \varepsilon_{15} = \varepsilon_{16} = 10^{-4}$, $\varepsilon_{10} = 10^{-2}$; (2) the population size $n = 100$; (3) $M_1 = 10^9$ and $M_2 = 10^7$ in the penalty function; (4) in MOIRCGA, the mutation probability $P_m = 0.5$ and the crossover probability $P_c = 1$; (5) in IRCGA-1, IRCGA-2, IRCGA-3, SPXMOGA, and HGA, the parameter settings are the same as in [27, 31, 37, 42, 50]; (6) reserve 50 elite individuals in MOIRCGA; and (7) the ES of IRCGA-1, IRCGA-2, IRCGA-3, and SPXMOGA are the same as that of in [27].

Input: population size: n; mutation probability: P_m; crossover probability: P_c; lower and upper bounds of variables: $[a, b]$; the number of elite individuals retained: m
Output: the optimal value of an optimization problem
(1) Initialize population
(2) Calculate the objective function values of individuals and sort the individuals in descending order
(3) While termination criterion is not met
(4) Perform SGS
(5) Perform HNDDBX
(6) Perform substitution operation
(7) Calculate the objective function values of individuals and sort the individuals in descending order
(8) Retain m elitist members from all parents and $3n$ offsprings after performing the substitution operation
(9) Perform combinational mutation
(10) Calculate the objective function values of the individuals and sort the individuals in descending order
(11) m elitist members, $3nP_m$ mutated members, and $3n(1-P_m)$ nonmutated members constitute the current population
(12) Select n best members to constitute the new population, and retain m elitist members
(13) End While
(14) Output the average running time, the average number of iterations, and the optimal value

Algorithm 1: MOIRCGA.

4.4. Comparison with the Results of Other Studies in the Literature. 16 test functions in Appendix A were used as examples and each test function was run on the same computer for 1000 times. The initial population of the various algorithms is the same.

All of the algorithms in this paper were developed in *Matlab R*2018*b* programming language. IRCGA-1 [27] proposed a direction-based heuristic crossover operator which is also used in this paper. IRCGA-2 [31] gives a selection method and a direction-based heuristic crossover operator. IRCGA-3 [37] gives a heuristic normal distribution crossover (HNDX) operator. However, the HNDX does not consider the influence of the optimal individual in the population on the offsprings. SPXMOGA [42] proposes a single-point crossover multi-offspring GA based on binary coding and gives the generation method of multi-offspring and the corresponding evolutionary strategy, where the crossover operator is not heuristic. A new hybrid GSA-GA algorithm is presented in [50], and a HNDX operator is used in program. All of the above algorithms have parallelism, and regardless of whether the problem to be optimized has a derivative, it can be solved by the above algorithms. The computational results of various algorithms are shown in Table 2.

The average running time and the average number of iterations at convergence to the optimal solution were calculated as follows: when the iteration termination condition is satisfied, the numbers of iterations and time of processing at the i-th run are, respectively, iter(i) and $t(i)$, $i = 1, 2, \ldots, k$, where k is the number of runs used in each experiment. Then, the average running time and the average number of iterations are computed by

$$t_{\text{ave}} = \frac{1}{k}\sum_{i=1}^{k} t(i),$$
$$\text{iter}_{\text{ave}} = \frac{1}{k}\sum_{i=1}^{k} \text{iter}(i), \tag{15}$$

where t_{ave} is the average running time and iter_{ave} is the average number of iterations.

When the average running time of ICGA-1 is t_1, the average running time of MOIRCGA is t_2; MOIRCGA reduces the average running time by x% for the i-th test function f_i in comparison to IRCGA-1. x% is computed by

$$t_2(1 - x\%) = t_1, \tag{16}$$

$$x = 100\left(1 - \frac{t_1}{t_2}\right). \tag{17}$$

The computational method of x% is also the same as equations (16) and (17) for IRCGA-2, IRCGA-3, SPXMOGA, and HGA.

It is observed in Table 2 that the average running time and the average number of iterations of MOIRCGA are significantly superior to those of IRCGA-1, IRCGA-2, IRCGA-3, SPXMOGA, and HGA. MOIRCGA reduces the average running time by 86.4723%, 78.1382%, 77.1723%, 92.3419%, and 7.7381% for f_1, 92.9825%, 81.4815%, 68.7500%, 94.8586%, and 51.2195% for f_2, 92.5178%, 33.6842%, 47.5000%, 92.3544%, and 7.3529% for f_3, 92.9012%, 36.1111%, 47.7273%, 96.2388%, and 65.9259% for f_4, 76.2048%, 15.0538%, 16.8421%, 88.0030%, and 11.7318% for f_5, 73.8462%, 56.4103%, 51.7730%, 74.9077%, and 88.6855% for f_6, 84.8276%, 85.3982%, 84.6154%, 89.7727%, and 77.0302% for f_7, 60.3208%, 55.6854%, 45.9896%, 81.7335%, and 41.9980% for f_8, 93.1741%, 78.0220%, 66.3866%, 93.9668%, and 54.7170% for f_9, 20.0993%, 56.8114%, 51.5437%, 62.8427%, and 15.8803% for f_{10}, 70.5069%, 58.9744%, 53.6232%, 96.5720%, and 53.6232% for f_{11}, 72.1264%, 70.2910%, 44.5714%, 91.5725%, and 31.4488% for f_{12}, 91.5225%, 37.7119%, 12.5000%, 92.3676%, and 5.7692% for f_{13}, 81.4116%, 15.8228%, 7.3171%, 87.5468%, and 7.9585% for f_{14}, 79.3843%, 85.8515%, 78.6885%, 98.8135%, and 64.6965% for f_{15}, and 48.7271%, 51.5362%, 46.5960%, 86.8151%, and 16.0934% for f_{16} in comparison to

Table 2: The computational results of various algorithms.

Function	IRCGA-1		IRCGA-2		IRCGA-3		SPXMOGA		HGA		MOIRCGA	
	t_{ave} (s)	$iter_{ave}$ (times)	t_{ave} (s)	$iter_{ave}$ (times)	t_{ave} (s)	$iter_{ave}$ (times)	t_{ave} (s)	$iter_{ave}$ (times)	t_{ave} (s)	$iter_{ave}$ (times)	t_{ave} (s)	$iter_{ave}$ (times)
f_1	0.1142	89.6250	0.0709	68.1900	0.0679	51.3940	0.2024	624.2430	0.0168	17.1530	0.0155	6.2150
f_2	0.0285	18.2660	0.0108	11.9562	0.0064	5.6540	0.0389	15.8500	0.0041	3.7000	0.0020	2.0330
f_3	0.0842	36.2200	0.0095	13.6253	0.0120	15.7290	0.0824	20.5700	0.0068	4.8800	0.0063	3.1400
f_4	0.0648	22.7620	0.0072	10.6824	0.0088	12.6810	0.1223	275.3800	0.0135	8.4500	0.0046	3.5120
f_5	0.0664	26.5900	0.0186	36.2560	0.0190	38.2210	0.1317	428.9000	0.0179	23.6320	0.0158	20.8310
f_6	0.0260	15.7500	0.0156	20.2610	0.0141	18.6410	0.0271	82.4800	0.0601	26.0600	0.0068	4.9460
f_7	0.1305	41.1900	0.1356	242.6236	0.1287	224.1050	0.1936	263.3200	0.0862	36.6340	0.0198	22.6480
f_8	0.2868	185.0700	0.2568	263.6246	0.2107	451.7110	0.6230	1785.6240	0.1962	156.2300	0.1138	127.0690
f_9	0.1758	59.6400	0.0546	80.6528	0.0357	60.8970	0.1989	96.7800	0.0265	16.5600	0.0120	12.7330
f_{10}	16.5687	11835.8600	30.6528	31042.6540	27.3205	28412.8500	35.6283	32461.6280	15.7377	10983.8614	13.2385	8414.4000
f_{11}	0.0217	25.8100	0.0156	32.6246	0.0138	28.1210	0.1867	42.8300	0.0138	8.5320	0.0064	6.9180
f_{12}	0.0696	50.5600	0.0653	86.6810	0.0350	56.0080	0.2302	127.5760	0.0283	21.6230	0.0194	19.5300
f_{13}	0.1734	58.6100	0.0236	26.0625	0.0168	11.0350	0.1926	72.9426	0.0156	9.6230	0.0147	7.8780
f_{14}	0.1431	46.7300	0.0316	65.2681	0.0287	49.7220	0.2136	98.9260	0.0289	33.2310	0.0266	31.3050
f_{15}	0.1072	34.0400	0.1562	226.3584	0.1037	180.7190	1.8626	236.2580	0.0626	28.5300	0.0221	24.9230
f_{16}	1.6890	1013.5681	1.7869	1926.5863	1.6216	1711.9600	6.5681	4267.8320	1.0321	1201.6560	0.8660	813.0560

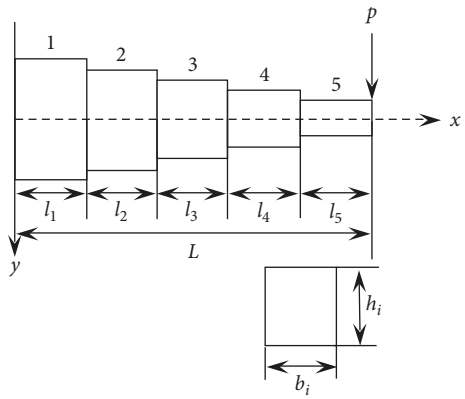


Figure 6: Schematic of the cantilevered beam structure with its design variables.

IRCGA-1, IRCGA-2, IRCGA-3, SPXMOGA, and HGA. Similarly, the number of iterations are reduced by 93.0655%, 90.8858%, 87.9022%, 99.0044%, and 63.7673% for f_1, 88.8700%, 82.9963%, 64.0432%, 7.1735%, and 45.0541% for f_2, 91.3308%, 76.9546%, 80.0369%, 84.7351%, and 35.6557% for f_3, 84.5708%, 67.1235%, 72.3050%, 98.7247%, and 58.4379% for f_4, 21.6585%, 42.5447%, 45.4985%, 95.1432%, and 11.8526% for f_5, 68.5968%, 75.5886%, 73.4671%, 94.0034%, and 81.0207% for f_6, 45.0158%, 90.6654%, 89.8940%, 91.3991%, and 38.1776% for f_7, 31.3400%, 51.7993%, 71.8694%, 92.8838%, and 18.6654% for f_8, 78.6502%, 84.2126%, 79.0909%, 86.8434%, and 23.1099% for f_9, 28.9076%, 72.8941%, 70.3852%, 74.0789%, and 23.3931% for f_{10}, 73.1964%, 78.7951%, 75.3992%, 83.8478%, and 18.9170% for f_{11}, 61.3726%, 77.4691%, 65.1300%, 84.6915%, and 9.6795% for f_{12}, 86.5586%, 69.7727%, 28.6090%, 89.1997%, and 18.1336% for f_{13}, 33.0088%, 52.0363%, 37.0399%, 68.3551%, and 5.7958% for f_{14}, 26.7832%, 88.9896%, 86.2090%, 89.4509%, and 12.6428% for f_{15}, and 19.7828%, 57.7981%, 52.5073%, 80.9492%, and 32.3387% for f_{16}. Thus, MOIRCGA is observed to be superior to IRCGA-1, IRCGA-2, IRCGA-3, SPXMOGA, and HGA in terms of all the measures utilized.

5. Parameter Optimization of Cantilever Beam

To further verify the validity of the MOIRCGA, the cantilever beam design problem with discrete cross-sectional area given in [50] was chosen as an application. The cantilever beam structure is shown in Figure 6.

The goal of the cantilever beam design optimization problem is to determine the optimal combination of five different cross-sectional areas to minimize the volume of the cantilever beam. The design problem has 10 variables. They are the width and height of each cross section, h_i and b_i ($i = 1, 2, \ldots, 5$). An external force $p = 50{,}000$ N is applied at the free end of the cantilevered beam. The maximum allowable stress at the end of each section is $\sigma_{\max} = 14{,}000\ \mathrm{N/cm^2}$, the material elasticity modulus E is 200 Gap, the length of each section l_i ($i = 1, 2, \ldots, 5$) is 100 cm, and the maximum allowable deflection is $y_{\max} = 2.715$ cm. The height-to-width aspect ratio of each cross-section is restricted to be less than 20. Then, the mathematical model for the optimization of this problem is defined as follows:

$$X = [b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5]^T = [x_1, x_2, \cdots, x_{10}]^T,$$

$$\min \quad f(X) = 100\left(x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10}\right),$$

$$\text{s.t.} \quad \begin{cases} g_1(x) = 10.7143 - \dfrac{x_1x_2^2}{10^3} \le 0, \\ g_2(x) = 8.5714 - \dfrac{x_3x_4^2}{10^3} \le 0, \\ g_3(x) = 6.4286 - \dfrac{x_5x_6^2}{10^3} \le 0, \\ g_4(x) = 4.2957 - \dfrac{x_7x_8^2}{10^3} \le 0, \\ g_5(x) = 2.1428 - \dfrac{x_9x_{10}^2}{10^3} \le 0, \\ g_6(x) = 10^4 * \left(\dfrac{244}{x_1x_2^3} + \dfrac{148}{x_3x_4^3} + \dfrac{76}{x_5x_6^3} + \dfrac{28}{x_7x_8^3} + \dfrac{4}{x_9x_{10}^3}\right) \\ \quad -10.8611 \le 0, \\ g_7(x) = x_2 - 20x_1 \le 0, \\ g_8(x) = x_4 - 20x_3 \le 0, \\ g_9(x) = x_6 - 20x_5 \le 0, \\ g_{10}(x) = x_8 - 20x_7 \le 0, \\ g_{11}(x) = x_{10} - 20x_9 \le 0, \\ 1 \le x_i \le 5,\ i = 1, 3, 5, 7, 9;\ 30 \le x_i \le 65,\ i = 2, 4, 6, 8, 10. \end{cases} \tag{18}$$

There are 11 constraints in this problem. Among them, $g_1(x)$–$g_5(x)$ are related to the allowable stress constraints, $g_6(x)$ is the constraint regarding the allowable deflection, and $g_7(x)$–$g_{11}(x)$ are restrictions to the geometric shape of the cross-section. The optimization design problem in this paper is addressed by using the penalty function method given by equation (3). The parameter settings are as described in Section 4.3.

The optimization results of MOIRCGA are compared with the optimization results in reference [27, 31, 37, 42, 50, 51]. The optimization results (the best objective function values and design variable values) that are obtained with

Table 3: Optimization results of various algorithms for the cantilever beam problem.

Variables and objective function	MOIRCGA	RCGA in reference [50]	IRCGA-1	IRCGA-2	IRCGA-3	SPXMOGA	HGA
b_1	3.0530	3.0459	3.005	3.0602	3.0450	3.0300	3.0442
h_1	60.9997	60.8969	60.004	61.2010	60.8763	59.6231	60.8812
b_2	2.8062	2.8018	3.0051	2.8160	2.8023	3.2011	2.8022
h_2	56.1227	56.0168	55.10	56.3020	56.0430	55.2301	56.0432
b_3	2.5236	2.5251	2.601	2.6020	2.5253	2.5891	2.5253
h_3	50.4718	50.4643	50.10	50.6970	50.5041	50.0620	50.5041
b_4	2.2063	2.2252	2.31	2.2237	2.2210	2.3001	2.2210
h_4	44.1253	44.4745	45.46	44.2758	44.4170	45.5310	44.4167
b_5	1.7498	1.7678	1.79	1.7513	1.7500	1.7921	1.7499
h_5	34.9948	34.8462	35.004	35.0131	34.9950	34.5863	34.9949
$f(x)$	62968.18	63044.17	64387.29	63752.19	62984.70	65377.86	62980.39

Table 4: The constrained values of various algorithms for cantilever beam problem.

Constrained values	MOIRCGA	RCGA in reference [50]	IRCGA-1	IRCGA-2	IRCGA-3	SPXMOGA	HGA
$g_1(x)$	−0.6458	−0.5814	−0.1051	−0.7479	−0.5702	−0.0571	−0.5691
$g_2(x)$	−0.2675	−0.2205	−0.5521	−0.3551	−0.2301	−1.1931	−0.2299
$g_3(x)$	−0.0000	−0.0020	0.0999	−0.2590	−0.0126	−0.0602	−0.0126
$g_4(x)$	−0.0000	−0.1158	−0.4782	−0.0635	−0.0860	−0.4726	−0.0860
$g_5(x)$	−0.0000	−0.00370	−0.0505	−0.0041	−0.0003	−0.0009	−0.0002
$g_6(x)$	−0.0036	−0.00074	−0.0238	−0.2136	−0.0005	−0.1487	−0.0003
$g_7(x)$	−0.0603	−0.02241	−0.096	−0.0030	−0.0237	−0.9769	−0.0028
$g_8(x)$	−0.0013	−0.02071	−5.002	−0.0180	−0.0030	−8.7919	−0.0008
$g_9(x)$	−0.0002	−0.03804	−1.92	−1.3430	−0.0019	−1.7200	−0.0019
$g_{10}(x)$	−0.0007	−0.03109	−0.74	−0.1982	−0.0030	−0.4710	−0.0033
$g_{11}(x)$	−0.0012	−0.51026	−0.796	−0.0129	−0.0050	−1.2557	−0.0031

MOIRCGA and genetic algorithm (RCGA) in reference [27, 31, 37, 42, 50, 51] are listed in Table 3, and the constraints are listed in Table 4.

It is clear from Table 3 that the function value obtained with the proposed MOIRCGA is superior to that with RCGA in reference [27, 31, 37, 42, 50, 51]. At the same time, Table 4 reveals that the optimal solution obtained by the two methods satisfies all constraints.

6. Conclusions

The better individuals in the population are more likely to approach the global optimal solution. However, the two offsprings generated by the LX operator are symmetric with respect to the parental position, and the two offsprings are not necessarily located near the better individual of the two parents. The two offsprings generated by the SBX operator are located on the straight line connecting the two parents, and are in the vicinity of the two parents. The two offsprings generated by the Modified SBX-crossover (MSBX) operator are located on the straight line connecting the two parents, and are in the vicinity of the two parents. The distance between two offsprings and two parents can be adaptively controlled by parameter η, which overcomes the deficiency of adjusting distribution index η which SBX cannot adaptively adjust. The two offsprings generated by the HX operator are located on the straight line connecting the two parents and are in the vicinity of the parents with better fitness value. The offsprings generated by the PCX operators are not close to the center of the parents but close to the parents. The two offsprings generated by the BLX-α operator are located at the center of the two parents. Since the DBX operator is only able to explore $2^n - 1$ possible search directions, the search directions of DBX are limited; when the dimensionalities of the variables are small, the null vector solution is likely to be generated. Currently, the search direction is randomly generated, and the search direction is not instructive. HNDDBX overcomes the shortcomings of the above crossover operators. It not only considers the optimal individuals in the population, but also ensures that the offsprings generated by the HNDDBX operator are located near the better individual among the parents of participation crossover. Therefore, the convergence speed of the MOIRCGA using the HNDDBX operator is significantly improved.

In most previous GA algorithms, as iterations increase, the same individuals are likely to appear in the population. Therefore, it is possible that the two parents of participation crossover are the same. Under these circumstances, the crossover operation does not generate new individuals, that is, the crossover operation does not work, which affects the computational efficiency of GA and the ability to explore other extremal regions. To avoid this problem, the substitution operation is added after the crossover so that there is no duplication of the same individuals in the population. This improves the computational efficiency of MOIRCGA for quick convergence to the global optimal solution.

The local and global search abilities of different mutation operators are different. Some mutation operators have strong local search ability, and some have strong global search ability. A single mutation operator is difficult to take into account both local and global search abilities. This paper proposes a combinational mutation method which include three mutation operators, that is, Normal mutation operator, Cauchy mutation operator, and Lévy mutation operator. The Normal mutation operator has strong local search ability, and the Cauchy mutation operator has strong global search ability. The Lévy mutation operator performs local search in most cases and occasionally performs global search as well, but the global search ability of the Lévy mutation operator is superior to the Cauchy mutation operator, thus helping the algorithm avoid falling into local optimum.

The computational results with sixteen examples show that MOIRCGA has better performance than the other methods in references [27, 31, 37, 42] with respect to all the measures of performance considered.

As an example application, the optimization problem of the cantilevered beam structure is formulated, and MOIRCGA in the paper and RCGA in reference [52] are used to optimize the parameters of the cantilevered beam structure. The optimization results show that the function value obtained with MOIRCGA is superior to that obtained with RCGA.

Appendix

A. Test Functions

The 16 test functions are as follows:

(1) f_1:

$$\begin{aligned} &\min \quad f_1(x) = \sum_{i=1}^{20} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \\ &\text{s.t.} \quad -5.12 \le x_i \le 5.12, \quad i = 1, 2, \ldots, 20. \end{aligned} \tag{A.1}$$

The global optimal solution of f_1 is located at $x^* = (0, 0, \ldots, 0)$ with $f_1(x^*) = 0$.

(2) f_2:

$$\begin{aligned} &\max \quad f_2(x) = -x_1^2 - 2x_2^2 + 0.4\cos(3\pi x_1) + 0.6\cos(4\pi x_2), \\ &\text{s.t.} \quad -10 \le x_1, x_2 \le 10. \end{aligned} \tag{A.2}$$

The global optimal solution of f_2 is located at $x^* = (0, 0)$ with $f_2(x^*) = 1$.

(3) f_3:

$$\begin{aligned} &\min \quad f_3(x) = -\sin(x_1)\left[\sin\left(\frac{x_1^2}{\pi}\right)\right]^{20} - \sin(x_2)\left[\sin\left(\frac{2x_2^2}{\pi}\right)\right]^{20}, \\ &\text{s.t.} \quad 0 \le x_1, x_2 \le \pi. \end{aligned} \tag{A.3}$$

The global optimal solution of f_3 is located at $x^* = (2.0230, 2.0230)$ with $f_3(x^*) = -1.80130$.

(4) f_4:

$$\begin{aligned} &\min \quad f_4(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2, \\ &\text{s.t.} \quad -10 \le x_1, x_2 \le 10. \end{aligned} \tag{A.4}$$

The global optimal solution of f_4 is located at $x^* = (-0.0898, 0.7126)$ and $(0.0898, -0.7126)$ with $f_4(x^*) = -1.031628$.

(5) f_5:

$$\begin{aligned} &\min \quad f_5(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2, \\ &\text{s.t.} \quad -10 \le x_i \le 10, \quad i = 1, 2. \end{aligned} \tag{A.5}$$

The global optimal solution of f_5 is located at $x^* = (1, 1)$ with $f_5(x*) = 0$.

(6) f_6:

$$\begin{aligned} &\min \quad f_6(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}, \\ &\text{s.t.} \quad \begin{cases} x_2 - x_1^2 \ge 1, \\ x_1 - \left(x_2 - 4\right)^2 \ge 1, \\ -10 \le x_1, x_2 \le 10. \end{cases} \end{aligned} \tag{A.6}$$

The global optimal solution of f_6 is located at $x^* = (1.2279723, 4.2453733)$ with $f_6(x^*) = -0.095825$.

(7) f_7:

$$\begin{aligned} &\min \quad f_7(x) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2, \\ &\text{s.t.} \quad \begin{cases} \left(x_1 - 0.05\right)^2 + \left(x_2 - 2.5\right)^2 - 4.84 \le 0, \\ 4.84 - x_1^2 - \left(x_2 - 2.5\right)^2 \le 0, \\ 0 \le x_1, x_2 \le 6. \end{cases} \end{aligned} \tag{A.7}$$

The global optimal solution of f_7 is located at $x^* = (2.246826, 2.381865)$ with $f_7(x^*) = 13.59084$.

(8) f_8:

$$\begin{aligned} &\min \quad f_8(x) = \left(x_1 - 10\right)^3 + \left(x_2 - 20\right)^3, \\ &\text{s.t.} \quad \begin{cases} -\left(x_1 - 5\right)^2 - \left(x_2 - 5\right)^2 + 100 \le 0, \\ \left(x_1 - 6\right)^2 + \left(x_2 - 5\right)^2 - 82.81 \le 0, \\ 13 \le x_1 \le 100, \\ 0 \le x_2 \le 100. \end{cases} \end{aligned} \tag{A.8}$$

The global optimal solution of f_8 is located at $x^* = (14.095, 0.84296)$ with $f_8(x^*) = -6961.8817$.

(9) f_9:

$$\begin{aligned} \min \quad & f_9(x) = (x_1 - 2)^2 + (x_2 - 1)^2, \\ \text{s.t.} \quad & \begin{cases} x_1 + x_2 - 2 \le 0, \\ x_1^2 - x_2 + 2 \le 0, \\ -5 \le x_1, x_2 \le 5. \end{cases} \end{aligned} \tag{A.9}$$

The global optimal solution of f_9 is located at $x^* = (0, 2)$ with $f_9(x^*) = 5$.

(10) f_{10}:

$$\begin{aligned} \min \quad & f_{10}(x) = 5\sum_{i=1}^{4} x_i + 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i, \\ \text{s.t.} \quad & \begin{cases} 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0, \\ 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0, \\ 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0, \\ -8x_1 + x_{10} \le 0, \\ -8x_2 + x_{11} \le 0, \\ -8x_3 + x_{12} \le 0, \\ -2x_4 - x_5 + x_{10} \le 0, \\ -2x_6 - x_7 + x_{11} \le 0, \\ -2 * x_8 - x_9 + x_{12} \le 0, \\ 0 \le x_{1,2,\cdots 9,13} \le 1, 0 \le x_{10,11,12} \le 100. \end{cases} \end{aligned} \tag{A.10}$$

The global optimal solution of f_{10} is located at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ with $f_{10}(x^*) = -15$.

(11) f_{11}:

$$\begin{aligned} \min \quad & f_{11}(x) = x_1^2 + x_2^2, \\ \text{s.t.} \quad & \begin{cases} 2.5 - x_1 - x_2 \ge 0, \\ x_2 - x_1^2 \ge 2, \\ -100 \le x_1, x_2 \le 100. \end{cases} \end{aligned} \tag{A.11}$$

The global optimal solution of f_{11} is located at $x^* = (0, 2)$ with $f_{11}(x^*) = 4$.

(12) f_{12}:

$$\begin{aligned} \min \quad & f_{12}(x) = -2x_1 + x_2 - x_3, \\ \text{s.t.} \quad & \begin{cases} x_1 + x_2 + x_3 \le 4, \\ 3x_2 + x_3 \le 6, \\ 24 - 20x_1 + 9x_2 - 13x_3 + 4x_1^2 - 4x_1x_2 + 2x_1x_3 \\ \quad +2x_2^2 - 2x_2x_3 + 2x_3x_1 + 2x_3^2 \ge 0, \\ 0 \le x_1, x_2 \le 2, \\ 0 \le x_3 \le 3. \end{cases} \end{aligned} \tag{A.12}$$

The global optimal solution of f_{12} is located at $x^* = (2, 0, 0)$ with $f_{12}(x^*) = -4$.

(13) f_{13}:

$$\begin{aligned} \min \quad & f_{13}(x) = \textstyle\sum_{i=1}^{20} x_i^2, \\ \text{s.t.} \quad & -5.12 \le x_i \le 5.12, \quad i = 1, 2, \ldots, 20. \end{aligned} \tag{A.13}$$

The global optimal solution of f_{13} is located at $x^* = (0, 0, \ldots, 0)$ with $f_{13}(x^*) = 0$.

(14) f_{14}:

$$\begin{aligned} \min \quad & f_{14}(x) = -x_1 - x_2, \\ \text{s.t.} \quad & \begin{cases} x_1x_2 \le 4, \\ 0 \le x_1 \le 4, \\ 0 \le x_2 \le 8, \end{cases} \end{aligned} \tag{A.14}$$

The global optimal solution of f_{14} is located at $x^* = (0.5, 8)$ with $f_{14}(x^*) = -8.5$.

(15) f_{15}:

$$\begin{aligned} \min \quad & f_{15}(x) = x_1 + x_2, \\ \text{s.t.} \quad & \begin{cases} x_1^2 + x_2^2 \le 4, \\ x_1^2 + x_2^2 \ge 1, \\ x_1 - x_2 \le 1, \\ -x_1 + x_2 \le 1, \\ -2 \le x_1, x_2 \le 2. \end{cases} \end{aligned} \tag{A.15}$$

The global optimal solution of f_{15} is located at $x^* = (-1.414215754, -1.414211371)$ with $f_{15}(x^*) = -2.828427125$.

(16) f_{16}:

$$\begin{aligned}
\min \quad & f_{16}(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141, \\
\text{s.t.} \quad & \begin{cases}
85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0, \\
-85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0, \\
80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0, \\
-80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0, \\
9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0, \\
-9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0, \\
78 \le x_1 \le 102, \\
33 \le x_2 \le 45, \\
27 \le x_i \le 45, \quad i = 3, 4, 5.
\end{cases}
\end{aligned} \tag{A.16}$$

The global optimal solution of f_{16} is located at $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ with $f_{16}(x^*) = -30665.539$.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the project of National Key R&D plan (grant no. 2018YFD0300105).

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan, Ann Arbor, MI, USA, 1975.

[2] S. Zheng, J. Lai, G. Liu et al., "An improved real-coded hybrid genetic algorithm," *Computer Applications*, vol. 26, no. 8, pp. 1959–1962, 2006, https://wenku.baidu.com/view/da6a390c844769eae009edbf.html?re=view.

[3] H. Liu, C. Cui, and J. Chen, "An improved genetic algorithm for solving traveling salesman problem," *Journal of Beijing Institute of Technology*, vol. 33, no. 4, pp. 390–393, 2013, http://journal.bit.edu.cn/zr/ch/reader/create_pdf.aspx?file_no=20130413.

[4] J. An and H. Jin, "Application of adaptive strategy in real-coded genetic algorithm," *Microelectronics and Computers*, vol. 28, no. 4, pp. 140–142, 2011, http://www.ixueshu.com/document/cfb21286006ff07b318947a18e7f9386.html.

[5] R. Qi, F. Qian, W. Du et al., "Multi-objective genetic algorithm based on elitist selection and individual migration," *Control and Decision*, vol. 22, no. 2, pp. 164–168, 2007, http://www.ixueshu.com/document/8f0dee7b4fffcbdd.html.

[6] K. A. de Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[7] Q. Liu, X. Wang, Q. Fu et al., "Collaborative coevolutionary genetic algorithm," *Journal of Software*, vol. 20, no. 4, pp. 765–775, 2012, http://www.ixueshu.com/document/a6a052536c12b09b318947a18e7f9386.html.

[8] S. Yu and S. Kuang, "Martingale analysis of convergence and convergence rate of elite genetic algorithm," *Control Theory and Applications*, vol. 27, no. 7, pp. 843–848, 2010, http://www.ixueshu.com/document/fc8b215a622f956e318947a18e7f9386.html.

[9] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Choice Reviews Online*, vol. 27, no. 2, pp. 27–0936, 1989.

[10] S. N. Ripon, S. Kwong, and K. F. Man, "A Real coding implementation of jumping gene genetic algorithm (RJGGA) for multiobjective optimization," *Information Sciences*, vol. 177, no. 2, pp. 632–654, 2007.

[11] C. B. Lucasius and G. Kateman, "Applications of genetic algorithms in chemometrics," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 170–176, Morgan Kaufmann, Los Altos, CA, USA, 1989.

[12] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 61–69, Morgan Kaufmann, San Francisco, CA, USA, 1989.

[13] Q. Liu, X. Wang, Q. Fu et al., "Double elite Co-evolutionary genetic algorithm," *Journal of Software*, vol. 23, no. 4, pp. 765–775, 2012, http://www.ixueshu.com/document/a6a052536c12b09b318947a18e7f9386.html.

[14] S. Yu and S. Kuang, "Convergence and convergence rate analysis of elitist genetic algorithm based on martingale approach," *Control Theory & Applications*, vol. 27, no. 7, pp. 843–848, 2010, http://www.ixueshu.com/document/fc8b215a622f956e318947a18e7f9386.html.

[15] C. T. Chen, C. K. Wu, and C. Hwang, "Optimal design and control of CPU heat sink processes," *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, vol. 31, pp. 184–195, 2008, http://ukacc.group.shef.ac.uk/proceedings/control2006/papers/f81.pdf.

[16] C. T. Chen and Y. C. Chuang, "An intelligent run-to-run control strategy for chemical–mechanical polishing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 23, pp. 109–120, 2010.

[17] J. D. Dyer, R. J. Hartfield, G. V. Dozier, and J. E. Burkhalter, "Aerospace design optimization using a steady state real-coded genetic algorithm," *Applied Mathematics and Computation*, vol. 218, pp. 4710–4730, 2012.

[18] C. W. Tasi, C. L. Lin, and C. H. Huang, "Microbrushless DC motor control design based on real-coded structural genetic algorithm," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 1, pp. 151–159, 2011.

[19] K. Valarmathi, D. Devaraj, and T. K. Radhakrishnan, "Real-coded genetic algorithm for system identification and controller tuning," *Applied Mathematical Modelling*, vol. 33, no. 8, pp. 3392–3401, 2009.

[20] M. K. Amjad, S. I. Butt, R. Kousar et al., "Recent research trends in genetic algorithm based flexible job shop scheduling problems," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–32, 2018.

[21] X. W. Zheng, Y. Li, H. Liu et al., "A study on a cooperative character modeling based on an improved NSGA II," *Multimedia Tools and Applications*, vol. 75, no. 8, pp. 4305–4320, 2016.

[22] Z. F. Zhu and P. Y. Liu, "Feasibility research of text information filtering based on genetic algorithm," *Scientific Research and Essays*, vol. 5, no. 22, pp. 3405–3410, 2010.

[23] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundation of Genetic Algorithms 2*, L. D. Whitley, Ed., pp. 187–202, Morgan Kaufmann, San Mateo, CA, USA, 1993.

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

[25] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *International Journal of Approximate Reasoning*, vol. 12, no. 3-4, pp. 299–315, 1995.

[26] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[27] Y. C. Chuang, C. T. Chen, and C. Hwang, "A simple and efficient real-coded genetic algorithm for constrained optimization," *Applied Soft Computing*, vol. 38, pp. 87–105, 2016.

[28] P. H. Tang and M. H. Tseng, "Adaptive directed mutation for real-coded genetic algorithms," *Applied Soft Computing*, vol. 13, no. 1, pp. 600–614, 2013.

[29] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real-coded genetic algorithms," *Proceedings of Conference on Genetic and Evolutionary Computing, GECCO-99*, Morgan Kaufmann, pp. 657–664, Morgan Kaufmann, San Mateo, CA, USA, 1999.

[30] P. Subbaraj, R. Rengaraj, and S. Salivahanan, "Enhancement of Self-adaptive real-coded genetic algorithm using Taguchi method for Economic dispatch problem," *Applied Soft Computing*, vol. 11, no. 1, pp. 83–92, 2011.

[31] Y. Y. Song, F. L. Wang, and X. X. Chen, "An improved genetic algorithm for numerical function optimization," *Applied Intelligence*, vol. 49, no. 5, pp. 1880–1902, 2019.

[32] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms I*, G. J. E. Rawlins, Ed., Morgan Kaufmann, San Mateo, CA, USA, 1991.

[33] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[34] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for functional optimization using unimodal normal distribution crossover," in *Proceedings of the 7th International Conference on Genetic Algorithms, ICGA-7*, T. Back, Ed., pp. 246–253, Morgan Kaufmann, East Lansing, MI, USA, 1997.

[35] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter evolution," *Evolutionary Computation*, vol. 10, no. 4, pp. 371–395, 2002.

[36] K. Deep and M. Thakur, "A new crossover operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 188, pp. 895–911, 2007.

[37] J. Q. Wang, Z. W. Cheng, O. K. Ersoy et al., "Improvement analysis and application of real-coded genetic algorithm for solving constrained optimization problems," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–16, 2018.

[38] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, New York, NY, USA, 1996.

[39] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Xplore, Perth, WA, Australia, 1995.

[40] D. Wang and S. C. Fang, "Just-In-Time production planning with Semi-infinite programming model and genetic algorithm," *Control and Decision*, vol. 11, no. 4, pp. 446–451, 1996, http://www.ixueshu.com/document/738ad652ae6307d7318947a18e7f9386.html.

[41] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014.

[42] J. Q. Wang, F. L. Wang, and H. X. Zhu, "Single-point crossover multi-offspring genetic algorithm," *Journal of Biomathematics*, vol. 30, no. 2, pp. 305–311, 2015.

[43] J. Q. Wang, O. K. Ersoy, M. Y. He et al., "Multi-offspring genetic algorithm and its application to the traveling salesman problem," *Applied Soft Computing*, vol. 43, pp. 415–423, 2016.

[44] F. L. Wang, X. M. Fu, H. X. Zhu et al., "Multi-child genetic algorithm based on two-point crossover," *Journal of Northeast Agricultural University*, vol. 47, no. 3, pp. 72–79, 2016.

[45] Y. Q. Hu, *Operations Research Tutorial*, Tsinghua University Press, Beijing, China, 5th edition, 2018.

[46] X. F. Xie, W. J. Zhang, and Z. L. Yang, "A parents selection strategy fighting premature convergence in floating genetic algorithms," *Control and Decision*, vol. 17, no. 5, pp. 625–628, 2002, http://www.wiomax.com/team/xie/paper/CCDC02.pdf.

[47] X. H. Wang, X. Y. Liu, and M. H. Bai, "Migration algorithm with Gaussian variation and steepest descent operator," *Journal of Computer Engineering and Applications*, vol. 45, no. 20, pp. 57–60, 2009.

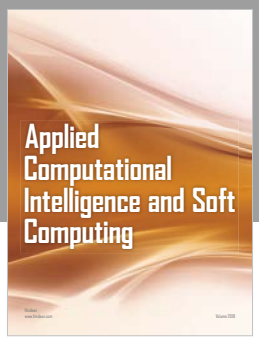
[48] J. J. Li and Y. M. Dai, "Frog Leaping algorithm based on adaptive mixed variation," *Journal of Computer Engineering and Applications*, vol. 49, no. 10, pp. 58–61, 2013.

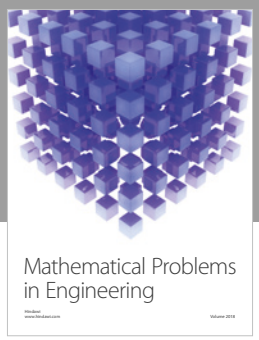
[49] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Physical Review E*, vol. 49, no. 5, pp. 4677–4683, 1994.

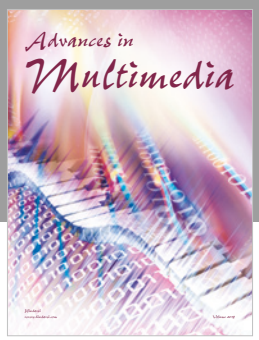
[50] H. Garg, "A hybrid GSA-GA algorithm for constrained optimization problems," *Information Sciences*, vol. 478, pp. 499–523, 2019.

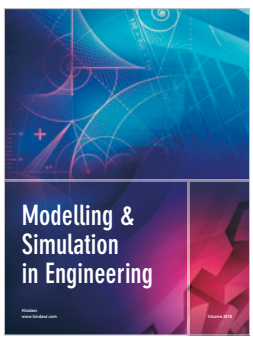
[51] H. C. Kuo and C. H. Lin, "A directed genetic algorithm for global optimization," *Applied Mathematics and Computation*, vol. 219, no. 14, pp. 7348–7364, 2013.

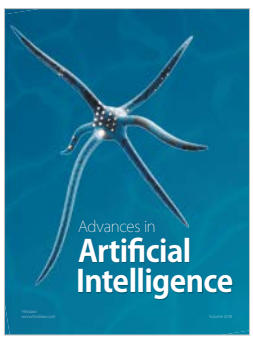
[52] J. Q. Wang, F. L. Wang, Z. G. Dong et al., "Research on evolutionary strategy of real-number genetic algorithm," *Mathematics in Practice and Theory*, vol. 47, no. 11, pp. 118–125, 2017.

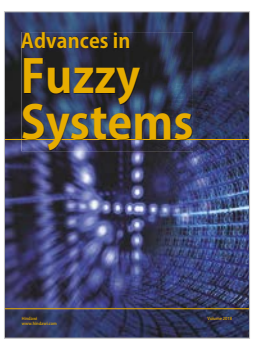


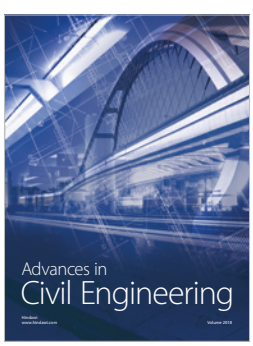


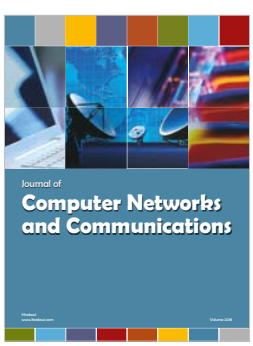


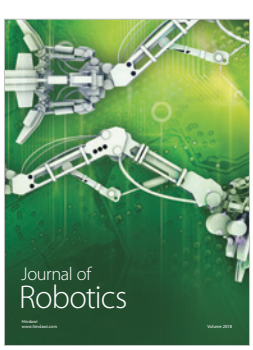


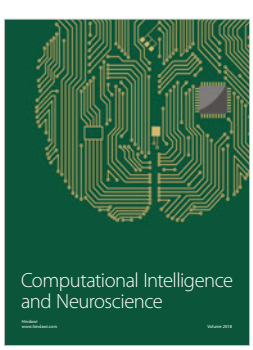












Hindawi

Submit your manuscripts at
www.hindawi.com