Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 6204728, 14 pages
http://dx.doi.org/10.1155/2016/6204728

Research Article

A Novel Hybrid Clonal Selection Algorithm with Combinatorial Recombination and Modified Hypermutation Operators for Global Optimization

Weiwei Zhang, Jingjing Lin, Honglei Jing, and Qiuwen Zhang

School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450000, China

Correspondence should be addressed to Weiwei Zhang; anqikeli@163.com

Received 12 May 2016; Accepted 31 July 2016

Academic Editor: Jens Christian Claussen

Artificial immune system is one of the most recently introduced intelligence methods which was inspired by biological immune system. Most immune system inspired algorithms are based on the clonal selection principle, known as clonal selection algorithms (CSAs). When coping with complex optimization problems with the characteristics of multimodality, high dimension, rotation, and composition, the traditional CSAs often suffer from the premature convergence and unsatisfied accuracy. To address these concerning issues, a recombination operator inspired by the biological combinatorial recombination is proposed at first. The recombination operator could generate the promising candidate solution to enhance search ability of the CSA by fusing the information from random chosen parents. Furthermore, a modified hypermutation operator is introduced to construct more promising and efficient candidate solutions. A set of 16 common used benchmark functions are adopted to test the effectiveness and efficiency of the recombination and hypermutation operators. The comparisons with classic CSA, CSA with recombination operator (RCSA), and CSA with recombination and modified hypermutation operator (RHCSA) demonstrate that the proposed algorithm significantly improves the performance of classic CSA. Moreover, comparison with the state-of-the-art algorithms shows that the proposed algorithm is quite competitive.

1. Introduction

Optimization techniques play a very important role in engineering design, commercial manufacture, financial market, information science, and related areas. An optimization problem can be expressed as

$$\begin{aligned} &\text{Optimize} \quad f(X), \\ &\text{subject to} \quad X \in \Omega, \end{aligned} \tag{1}$$

where $X = [x_1, x_2, \ldots, x_D]$ is a D-dimensional vector of decision variables in the feasible region Ω. The optimization could be either a minimization problem or a maximization problem. Traditional optimization algorithms often fail to deal with multimodal, nonconvex, nondifferentiable problems since most of them rely on the gradient information [1]. In the past few decades, inspired by nature, people have developed many optimization computation methods to solve the complicated optimization problems, including genetic algorithm (GA), differential evolution (DE), particle swarm optimization (PSO) artificial immune system (AIS), and some new nature-inspired algorithms [2–7].

Among them, artificial immune system (AIS) is a newly emerging computational paradigm inspired by the fundamentals of immune system. It abstracts the structure and function of the biological immune system and exploits the applications to solve computational problems. In the area of optimization, numerical comparisons demonstrated that the performance of AIS is competitive compared to that of the other nature-inspired algorithms [8]. Among them, clonal selection algorithm is one of well-known AIS and has been applied to solve many practical optimization problems [9–11]. The clonal selection algorithm is inspired from behavior of B cells in secreting antibody to bind with the invading antigen. In view of the good performance of clonal selection algorithms (CSAs) in computational optimization area [12], in this paper, our work concentrates on CSA.

CSAs have attracted a lot of attention since they were developed [13–15]. However, with the increase of the complexity of the optimization problems, the deficiency of CSA is gradually exposed. One of the main shortages is that hypermutation is the main operator to modify the construction of the candidate solution in the traditional CSAs, and generally both global search and local search are based on adjusting the step size of the hypermutation. It would be barely unsatisfactory on balancing the diversity and convergence. In some improved versions, new randomly generated solutions are introduced by receptor editing or other mechanisms to increase diversity. However, the mechanism of randomly introduced solutions, along with the step-size controlled hypermutation leads to a blind optima searching process. Thereafter, premature convergence and diversity loss happen when dealing with the high-dimensional, multimodal optimization problems.

In view of the above analysis, we are motivated to explore the undeveloped potential of clonal selection theory in this paper. Inspired by the immune response of B cells, a combinatorial recombination operator is introduced to share the responsibility with hypermutation. In the proposed algorithm, recombination operator is proposed to enhance the search ability of the CSA. Moreover, the hypermutation operator is modified to generate more promising candidate solutions.

The rest of this paper is organized as follows. In Section 2, the general framework of CSA algorithm is presented, and the research works on improved CSAs in the field of numerical optimization are reviewed. In Section 3, the RHCSA algorithm is proposed based on the new introduced recombination and the modified hypermutation operators. Section 4 presents and discusses the experimental results. Finally, the conclusion is drawn in Section 5.

2. CSA Framework

The AIS has become popular from the late 1990s. Several books, journals, and conference papers have been published in the past few years. de Castro and Timmis [16] depict the main models in AIS such as clonal selection, immune networks, and negative selection theories. More detailed review of AIS and their applications can be found in [15, 17–19]. Here we will give a brief literature review of the contemporary research efforts in clonal selection based algorithms in handling the numerical optimization.

2.1. Clonal Selection Algorithms (CSAs). The main idea of clonal selection theory lies in the phenomenon where B cell reacts to invaded antigen through modifying the receptor called antibody. The general one, named CLOGNALG [20], is one of the representatives for clonal selection algorithms. There are mainly three operations involved, which are cloning, hypermutation, and selection.

The general framework of clonal selection algorithm for optimization is presented as follows.

Framework of CSA

Step 1 (initialization). Randomly initialize antibody population.

Step 2 (evaluation). Evaluate the objective values of the antibody population as their fitness.

Step 3 (cloning). Generate copies of the antibodies.

Step 4 (hypermutation). Mutate all the generated copies.

Step 5 (selection). Select the one with highest fitness to survive.

Step 6. Repeat Steps 2–5 until a termination criterion is met.

The main features of the CSA framework are (i) the clone number, usually proportionally to the affinity of antibody with respect to the antigens; (ii) the mutation rate, normally inversely proportional to the affinity; and (iii) the absence of recombination operators (such as crossover in GAs). Those characteristics expose deficiencies when facing the high-dimensional, nonconvex multimodal, and multiobjective optimization problems. First, it is obvious that the cloning operator consumes high computational resource. Second, hypermutation is insufficient to bear the burden of balancing both the global search and the local search, which will lead to the premature convergence and unsatisfied accuracy. Third, the lack of consideration on interaction among individuals in the population may lead to the search missing global awareness, that is, overly searching one or some areas of the search space while leaving the others unvisited.

2.2. Improved CSAs. To overcome the abovementioned shortcomings, a bunch of improved algorithms based on CSA are proposed. The research works can be briefly classified as three categories.

2.2.1. Modification of the Operators or Introduction of New Operators within the CSA Framework. Cutello et al. introduced a real-coded clonal selection algorithm for global optimization, involving the cloning operator, inversely proportional hypermutation, and aging operator [21]. Three versions of somatic contiguous hypermutation operator are analyzed and proven to have better performance than standard bit mutation to some types of optimization problem [22]. Khilwani et al. [23] proposed a fast clonal algorithm (FCA) which designs a parallel mutation operator comprising Gaussian and Cauchy mutation strategies. Lu and Yang [24] introduce the Cauchy mutation for the improved CSA (IMCSA). Two versions of immune algorithm named OPT-IMMALG01 and OPT-IMMALG combined with the clonal operator, M hypermutation operator, aging operator, and $(\mu + \lambda)$ selection operator based on binary-code and real-code representation, respectively, are discussed. The experimental results approve the effectiveness of the algorithms in handing the high-dimensional global numerical optimization problem [12]. Randomized clonal expansion strategy is proposed to solve high-dimensional global optimization problem in [25].

2.2.2. Combine CSA with Immune Network Theory. Immune network theory considers that the immune cells have relations with each other and hereafter the cells, molecules, and

other related substances construct a network. CAS combing with immune network theory is more suitable for handling multimodal function optimization.

The opt-AINet algorithm is an earlier version of CSA with artificial immune networks, which is proposed to solve multimodal continuous function optimization [26]. Uniform cloning operator, affinity-based Gaussian mutation, and similarity-based suppressor are proposed. To enhance the parameter adaptation, an improved adaptive Artificial Immune Network called IA-AIS is proposed, where affinity-based cloning operator, controlled affinity-based Gaussian mutation, and dynamic suppressor are introduced [27]. By imitating the social behaviors of animals, social leaning mechanism is introduced to the immune network; an algorithm called AINet-SL is proposed. The population is separated into elitist swarm (ES) and the common swarm (CS). Nonlinear affinity-based cloning, self-learning mutation, and social learning mutation are employed [28]. Based on the affinity measure, the candidate solution space is divided into the elitist space, the common space, and the poor space, and different hypermutation strategies are applied, respectively, in MPAINet [29]. Concentration-Based Artificial Immune Network, where the concentration of antibody is introduced to stimulate and maintain the diversity of the population, is applied to continuous optimization [30], combinatorial optimization [31], and multiobjective optimization [32].

2.2.3. Hybrid CSA. CSA is also hybrid with PSO, DE, and other evolutionary strategies. Hill-climbing local search operator is combined with immune algorithm in [33]. Differential immune clonal selection algorithm (DICSA) is put forward by Gong et al. [34], where the differential mutation and differential crossover operators are introduced. Orthogonal initialization and neighborhood orthogonal cloning operator are proposed in an orthogonal immune algorithm (OIA) [35]. CSA with nondominated neighborhood selection strategy (NNIA) [36] was proposed for multiobjective optimization. Shang et al. [37] proposed an immune clonal algorithm (NICA) for multiobjective optimization problems, which makes improvements on four aspects in comparison with the traditional clonal selection computing model.

Baldwin effect is introduced into CSA and formulated the Baldwinian clonal selection algorithm (BCSA), which guides the evolution of each antibody by the differential information of other antibodies in the population [38]. Gong et al. [39] substitute the mutation in CSA with the local search technique in *Lamarckian* Clonal Selection Algorithm (LCSA) and adopt recombination operator and tournament selection operator for numerical optimization. An enhanced CSA, hybrid learning CSA (HLCSA) [40], is proposed by introducing two learning mechanisms: Baldwinian learning and orthogonal learning.

Although many improvements on immune inspired algorithm have been realized, the abovementioned shortcoming of artificial immune algorithm, such as premature convergence, high computational cost, and unsatisfied accuracy that can negatively affect the application of immune algorithm, remains a problem. The search ability of immune algorithm is limited when dealing with high-dimensional complex optimization with different characteristics. In this paper, we propose an improved CSA, by introducing a recombination operator and modifying the hypermutation operator to cope with the complex optimization problems with the characteristic high-dimensional, nonconvex, rotated, and composed optimization problems.

3. CSA with Combinatorial Recombination and Modified Hypermutation

3.1. Combinatorial Recombination. In immunology, the presence of both recombination and somatic mutation takes the responsibility for the diversification of antibody genes. The recombination of the immunoglobulin gene segments is the first step when the cells are first exposed to antigen. In the past few decades, recombination is mentioned more as crossover in GA. However, as depicted in [41], the recombination of immunoglobulin genes involved in the production of antibodies differs from the recombination (crossover) of parental genes in sexual reproduction. In the former, nucleotides can be inserted and deleted randomly from the recombined gene segments, while in the latter the genetic mixture is generated from parental chromosomes.

The basic unit of antibody molecule has a Y-shape structure, which contains two identical light chains and two identical heavy chains as shown in Figure 1(a). Variable regions located at the tips of the Y are primarily responsible for antigen recognition. Within these variable regions, some polypeptide segments show exceptional variability. The antibody molecules can be synthesized by an individual lying in the way of encoding the amino acid sequences of the variable domains into DNA chains as well as the random selection and recombination of gene segments as shown in Figure 1(b). During the development of B cell, the gene segments in the libraries are combined and rearranged at the level of the DNA. With the recombination of gene segments, recombination creates a population of cells that vary widely in their specificity. In this way, few immune cells are compatible with various antigens. After altering the base of antibody, the mutations fine-tune the lymphocyte receptor to better match the antigen [41, 42].

In the perspective of optimization, recombination functions as the coarse-grained exploration while hypermutation works the same way as fine-grained exploitation. Inspired by this, a combinatorial recombination operator is proposed as follows.

To avoid the truncation error and the complexity of the coding, our algorithm is coded in real number and each dimension of a solution is viewed as a gene segment. The whole population forms the gene fragments library. According to the recombination in immunology, any orderly rearrangement of gene segments would generate a new B cell. As presented in Figure 2, the recombination could be (a) between two individuals as crossover or (b) among several individuals as the combination of randomly selected gene segments. With the help of normalization, the combination of gene segments could be in the specific order, such as in Figure 2(a) or can be randomly arranged as shown in Figure 2(b) in the computational respective.

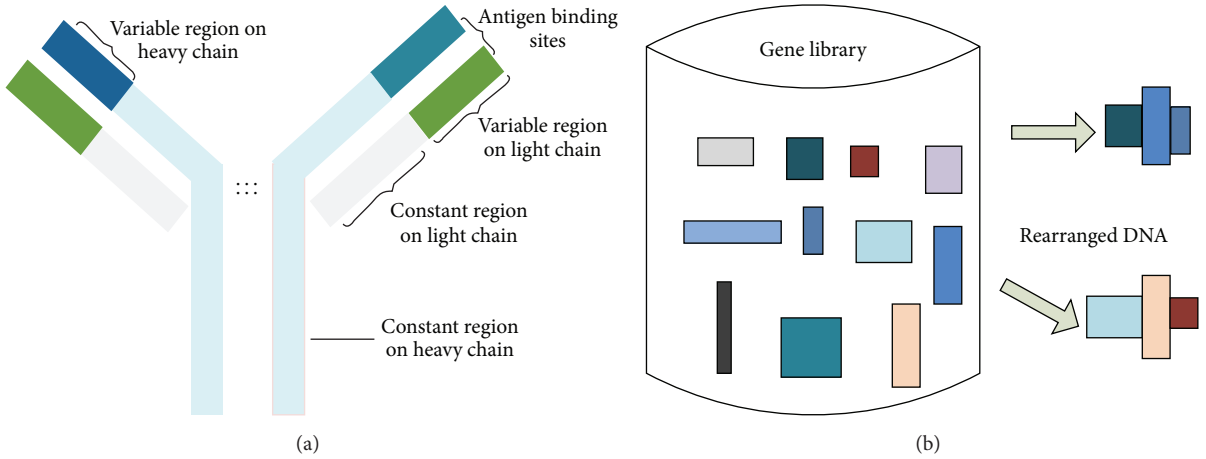

Figure 1: Structure of antibody (a) and recombination (b) in the variable region.

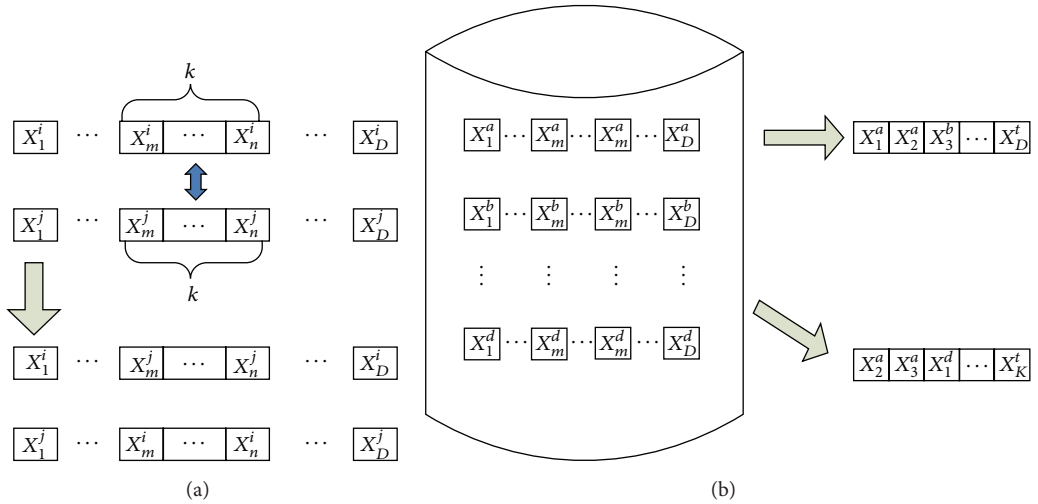

Figure 2: The way of rearrangement of gene segments (a) between two individuals (b) among several individuals.

The former one is similar to the SBX recombination in GA [42], where the crossover of parents could swap gene segments on one site or many sites. To better simulate the arrangement of gene segments, line recombination, DE inspired recombination [43], and intelligent recombination [44] are proposed. There would be a lot of ways to do the combinations with optional joined gene segments. With the randomness of the arrangements and combination, diversity is introduced. However, as is known to all, too much introduced diversity would be harmful for the performance. Based on the experimental experience, our work focuses on the way to process recombination between two parents. A new combinatorial recombination operator combining with line recombination is proposed as follows.

Randomly choose two individuals from the population, denoted by X_A and X_B, and then randomly choose m dimensions, $m \in [1, D]$, from each of them, where dimensions index could be recorded as vectors V_A and V_B, respectively. The new individuals are generated by

$$X_A'^{V_A} = \alpha X_A^{V_A} + (1-\alpha) X_B^{V_B}$$
$$X_B'^{V_B} = \alpha X_B^{V_B} + (1-\alpha) X_A^{V_A}, \quad (2)$$

where α is a randomly produced number between 0 and 1. It should be noted that the range of each dimension of decision variable should be normalized at first. The new proposed recombination operator is represented in Figure 3.

As shown in Figure 3, two new individuals are generated through the combinational recombination. In the example, m equals 3. It needs to be known that i_1 could be different from j_1, the same as in i_2 and j_2 and i_3 and j_3 as long as the normalization has been done.

Then, the fitness of the new generated individuals is evaluated. Together with the original individuals, two with

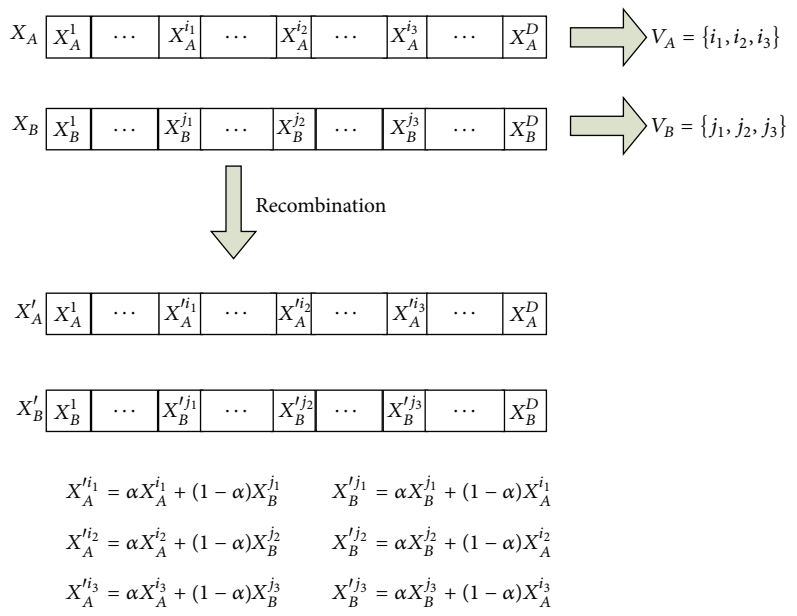

Figure 3: Recombination process.

the higher fitness will survive, and the other two individuals are deleted; that is, choose two individuals with high fitness from the set $\{X_A, X_B, X'_A, X'_B\}$. Instead of comparing with the whole population and reserving the elite, a better diversity will be maintained this way.

3.2. Modified Hypermutation. Hypermutation operator brings diversity for the population by introducing perturbation for each clone. Although there are several ways to implement this operator [22], inversely proportional strategy is always the main basis. The concept of the operator proposed in [12] is adopted in this research work, where each candidate solution is subject to M mutations without explicitly using a mutation probability. The inversely proportional law is used to determine the number of the mutations M:

$$\begin{aligned} \alpha &= \exp\left(-\rho f^*\left(X_i\right)\right) \\ M &= \lfloor (\alpha \times n) + 1 \rfloor, \end{aligned} \tag{3}$$

where $f^*(X_i) \in [0, 1]$ is the normalized fitness of X_i, ρ is the decay constant which determines the shape of the mutation rate, and $\lfloor \cdot \rfloor$ returns the lower bound integer. Then, M mutation is performed on each candidate solution:

$$X'^{j}_i = \begin{cases} X^j_{r1} + \lambda\left(X^j_{r1} - X^j_{r2}\right) & \text{if } j \in \text{rand } M(n) \\ X^j_i & \text{otherwise.} \end{cases} \tag{4}$$

$X_i(j)$ is the jth dimension of the ith individual, rand $M(n) \in \{1, \ldots, n\}$ is randomly chosen M indexes without repetition, and λ is a random number in the range of $[-1, 1]$. $r1, r2 \in \{1, 2, \ldots, N\}$ are randomly selected numbers; hereafter, the amplitude of the hypermutation is controlled automatically by the difference of randomly selected individuals in the population. The mutation equation (4) could be considered as the variant of differential evolution and is introduced by [45] to modify the original updating food source of artificial bee colony algorithm (ABC), where it is improved to benefit for enhancing the search ability of ABC. In our proposed algorithm, the equation is combined with the hypermutation inversely proportional M strategy as depicted above. The M strategy controls the direction, that is, along how many dimensions, while the equation controls the distance of the mutated clones with their parents. With combination of both, the amplitude of the hypermutation is automatically controlled according to the distribution of the population.

3.3. Framework of the Proposed Algorithm. Combined with the proposed combinatorial recombination and hypermutation operator, the framework of the proposed algorithm is as follows.

Step 1 (initialization). Randomly initialize a population Ab of N individuals, where N denotes the size of the initial population. Each initial solution $X_i = \{X_i(1), X_i(2), \ldots, X_i(D)\}$ is produced randomly within the range of boundaries of the decision space:

$$X^j_i = X^j_{\min} + \text{rand } (0, 1)\left(X^j_{\max} - X^j_{\min}\right), \tag{5}$$

where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, where D is the dimension of the decision variables, and $X^j_{\min}$ and $X^j_{\max}$ are the lower and upper bounds for the dimension j, respectively.

Step 2 (evaluation). Evaluate the objective value of each solution as its fitness.

Step 3 (recombination). Randomly choose two individuals from the population and implement the combinatorial

recombination as described in Section 3.1. The recombination rate is set to N_r.

Step 4 (clonal selection). (i) Cloning: each individual X_i generates N_c copies $\{X_i^1, X_i^2, \ldots, X_i^{N_c}\}$, where N_c is the clone number, which is a user defined constant.

(ii) Hypermutation: each clone X_i^j, $j = 1, \ldots, N_c$, goes through the hypermutation as described in Section 3.2 and generates the hypermutated clones $X_i'^j$.

(iii) Selection: select the individual with highest fitness among X_i and hypermutated clones $\{X_i'^1, X_i'^2, \ldots, X_i'^{N_c}\}$.

Step 5. If stopping condition is not met go to Step 3. Otherwise, output the best one of the feasible group.

It could be observed that instead of being proportional to the fitness, the clone number is a constant, which not only saves computational source but also avoids individuals being concentrated in some decision area and leaving the others unvisited when handling the multimodal problems.

4. Experimental Studies on Function Optimization Problems

In this section, experiments are conducted to evaluate the performance of RHCSA by using 16 commonly used global optimization benchmark problems. These functions contain the characteristics of being unimodal as $f_1 \sim f_2$, unrotated multimodal as $f_3 \sim f_8$, rotated multimodal as $f_9 \sim f_{14}$, and composite functions as $f_{15} \sim f_{16}$. Table 1 gives the expression of the benchmark function. The detailed characteristics of these functions can be found in [1, 46].

The introduced parameters are analyzed at first, and then the proposed algorithm is compared with traditional CSA in the benchmark functions. Finally, comparisons between the proposed algorithm and the state-of-the-art evolutionary computing models are represented. Some discussions on the performance analysis of RHCSA are also included. It needs to be noted that the same setup for the experiments is used for all the involved peers algorithms; that is, the comparison of all the algorithms is under the same configuration of experiment setup in this paper.

4.1. Experimental Parameters Settings. The experiments were conducted on 16 benchmark functions. The maximum number of function evaluations (mFES) is set as $10{,}000 * D$ for 10-D and 30-D situations, respectively. Each test is run 30 independent times.

There are quite few parameters introduced in the proposed algorithm. For a fair comparison among CSAs, they are tested using the same setting of the parameters, that is, the population size N is set to 30 and clone number N_c is set to 4 as in [40]. Furthermore, there are two new parameters introduced. One is m which controls the direction of combinatorial recombination, and the other is recombination rate. Tables 2 and 3 give the experimental results of varying m in the 16 benchmark functions with 10-D and 30-D. For convenience, the experiment sets 4 proportions to the dimension as the value to m, which are $[\lceil (1/10)D \rceil, \lceil (1/5)D \rceil, \lceil (1/3)D \rceil, D]$.

The recombination operator could introduce diversity to the population through fusing information between randomly chosen parents. When m is small, the generated offspring is much similar to one of the chosen parents, and only little dimension obtains information from both parents. When m is large, the generated offspring tend to be the fusion of chosen parents. That is to say, m controls the offspring being much like one of the chosen parents or the fusion of both of them. From Tables 2 and 3, we can find that the difference among the varying setup of m is quite small. In a more careful observation, $\lceil (1/5)D \rceil$ and $\lceil (1/3)D \rceil$ are more appropriate to compare with the other situations for both $D = 10$ and $D = 30$, and $\lceil (1/3)D \rceil$ is even better. It could be observed that experimental results are not sensitive to the parameter m, and a slightly larger m may be beneficial for the algorithm. In view of the observation, $m = \lceil (1/3)D \rceil$ is chosen in our experiments.

There is another parameter named recombination rate N_r, which controls the rate of recombination process, which will be implemented. The higher the N_r, the more recombination processes that will be executed. In the early stage, the population is uniformly distributed in the search space, and recombination of randomly chosen individuals will bring diversity to the population. As is known to all, diversity is beneficial for enhancing the search ability of the algorithm, but too much diversity will slow the convergence. Tables 4 and 5 represent the experimental results of varying the recombination rate.

From Tables 4 and 5, it could be observed that RHCSA has a quite robust performance with the varying recombination rate. By comparison, N_r which is equal to 0.7 achieves the best performance and is adopted in the paper.

4.2. Comparisons with Classic CSA. The experiments are implemented to check if the proposed operators are beneficial for the performance of the algorithm. Firstly, recombination operator is added to the original CSA denoted as RCSA, and then the modified hypermutation is introduced as RHCSA.

Table 6 shows the statistical results of CLONALG, RCSA, and RHCSA in optimizing the 16 test problems with $D = 10$ based on 30 independent runs including the mean and standard deviation. From the table, we can observe that, for all these test instances, the introduced recombination operator could obviously improve the performance of the CSA. The reason is that the recombination operator could bring diversity to enhance the search ability and avoid being trapped into the local optima. The experiments results on multimodal function $f_3 \sim f_8$ could present the ability of introduced recombination operator. With the modified hypermutation operator, the performance of the algorithm obtains a further improvement. This is because both recombination and hypermutation adopt the difference of chosen individuals to generate the candidate solution in the proposed algorithm. In the early stage, the population is distributed in the search space, and the difference is relatively large which is beneficial for global search and then, with the iterations going on, the population converges to the optima and the difference between individuals becomes smaller and smaller; that is, the search process turns to be a local search. It can

Table 1: Benchmark functions used in our experimental study.

Name	Test function	D	S	$f_{\min}$
Sphere function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	10/30	$[-100, 100]$	0
Rosenbrock's function	$f_2(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$	10/30	$[-2.048, 2.048]$	0
Ackley's function	$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\left(2\pi x_i\right)\right) + 20 + e$	10/30	$[-32.768, 32.768]$	0
Griewanks's function	$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\frac{x_i}{\sqrt{i}} + 1$	10/30	$[-600, 600]$	0
Weierstrass function	$f_5(x) = \sum_{i=1}^{D}\left\{\sum_{k=1}^{k\max}\left[a^k \cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right]\right\} - D\sum_{k=1}^{k\max}\left\{a^k \cos\left(2\pi b^k \cdot 0.5\right)\right\}$ $a = 0.5,\ b = 3,\ k\max = 20$	10/30	$[-0.5, 0.5]$	0
Rastrigin's function	$f_6(x) = \sum_{i=1}^{D}\{x_i^2 - 10\cos\left(2\pi x_i\right) + 10\}$	10/30	$[-5.12, 5.12]$	0
Noncont.Ras	$f_7(x) = \sum_{i=1}^{D}\{y_i^2 - 10\cos(2\pi y_i) + 10\},$ $y_i = \begin{cases} x_i & \lvert x_i\rvert < 0.5 \\ \frac{\text{round}\left(2x_i\right)}{2} & \lvert x_i\rvert \geq 0.5 \end{cases},\ i = 1, 2, \ldots, D$	10/30	$[-5.12, 5.12]$	0
Schwefel's function	$f_8(x) = 418.9829D - \sum_{i=1}^{D}\left\{x_i \sin\left(\lvert x_i\rvert^{0.5}\right)\right\}$	10/30	$[-500, 500]$	0
Rot.Ackley's function	$f_9(x) = f_3\left(y\right),\ y = M * x$	10/30	$[-32.768, 32.768]$	0
Rot.Griewanks's function	$f_{10}(x) = f_4\left(y\right),\ y = M * x$	10/30	$[-600, 600]$	0
Rot.Weierstrass function	$f_{11}(x) = f_5\left(y\right),\ y = M * x$	10/30	$[-0.5, 0.5]$	0
Rot.Rastrigin's function	$f_{12}(x) = f_6\left(y\right),\ y = M * x$	10/30	$[-5.12, 5.12]$	0
Rot.noncon Ras function	$f_{13}(x) = f_7\left(y\right),\ y = M * x$	10/30	$[-5.12, 5.12]$	0
Rot.Schwefel's function	$f_{14}(x) = 418.9829D - \sum_{i=1}^{D} Z_i,$ $Z_i = \begin{cases} y_i \sin\left(\lvert y_i\rvert^{0.5}\right) & \lvert y_i\rvert \leq 500 \\ 0.001\left(\lvert y_i\rvert - 500\right)^2 & \lvert y_i\rvert \geq 500 \end{cases},\ i = 1, 2, \ldots, D;$ $y = M * (x - 420.96) + 420.96$	10/30	$[-500, 500]$	0
Composition 1	$f_{15} = CF1$	10/30	$[-5, 5]$	0
Composition 2	$f_{16} = CF2$	10/30	$[-5, 5]$	0

be observed that RCSA surpass CSA and be surpassed by RHCSA at all the tested functions. It can be concluded that the proposed recombination and hypermutation operators are effective improving the ability of the CSA.

Table 7 shows the experimental results of the algorithms when $D = 30$. Similar results are represented, which indicates that RHCSA is able to handle the high-dimensional optimization problems as well.

4.3. Comparisons with the State-of-the-Art Algorithms. To compare RHCSA with the state-of-the-art algorithms, experimental results of seven representative evolutionary algorithms are listed in Tables 8 and 9. These algorithms are Baldwinian clonal selection algorithm (BCSA) [38]; hybrid learning clonal selection algorithm (HLCSA) [40]; orthogonal crossover based differential evolution (OXDE) [47]; self-adaptive differential evolution (SaDE) [48]; global and local

Table 2: Results (mean ± std) of RHCSA with varying sampling points of combinatorial recombination in 16 benchmark functions with $D = 10$.

Function	$m = \lceil (1/10)\, D \rceil$	$m = \lceil (1/5)\, D \rceil$	$m = \lceil (1/3)\, D \rceil$	$m = D$
f_1	1.3521e − 180 ± 0.0000e − 000	1.53784e − 166 ± 0.0000e − 000	**7.2585e − 195 ± 0.0000e − 000**	2.2773e − 078 ± 4.9085e − 078
f_2	8.6187e − 004 ± 3.20378e − 004	8.1377e − 004 ± 3.0271e − 004	8.9516e − 004 ± 3.2039e − 004	**2.8971e − 004 ± 3.6648e − 004**
f_3	5.2831e − 015 ± 4.2156e − 015	1.7763e − 015 ± 1.7763e − 015	**8.8817e − 016 ± 0.0000e − 000**	2.0724e − 015 ± 2.0511e − 015
f_4	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	4.0175e − 015 ± 1.3376e − 015	**3.0923e − 015 ± 1.4969e − 015**	3.9292e − 015 ± 1.1420e − 015	4.2912e − 015 ± 1.0827e − 015
f_{10}	9.3401e − 002 ± 1.5480e − 002	1.8195e − 002 ± 1.0245e − 002	**1.6937e − 002 ± 1.6091e − 002**	5.5463e − 002 ± 2.6635e − 002
f_{11}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{12}	2.4401e + 000 ± 1.0935e + 000	2.0556e + 000 ± 1.1601e + 000	**1.5954e + 000 ± 1.5954e + 000**	1.6638e + 000 ± 1.7026e + 000
f_{13}	8.1024e + 000 ± 5.6678e + 000	4.1385e + 000 ± 2.1123e + 000	**4.0000e + 000 ± 2.1213e + 000**	4.1773e + 000 ± 3.7530e + 000
f_{14}	7.5454e − 008 ± 6.2091e − 007	**6.8712e − 008 ± 7.0921e − 007**	9.8137e − 008 ± 7.7851e − 007	9.2565e − 008 ± 8.6635e − 007
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	9.0023e + 000 ± 7.8901e + 000	6.4971e + 000 ± 7.9297e + 000	**5.2613e + 000 ± 5.0573e + 000**	8.6977e + 000 ± 7.4630e + 000

Table 3: Results (mean ± std) of RHCSA with varying sampling points of combinatorial recombination in 16 benchmark functions with $D = 30$.

Function	$m = \lceil (1/10)\, D \rceil$	$m = \lceil (1/5)\, D \rceil$	$m = \lceil (1/3)\, D \rceil$	$m = D$
f_1	3.1129e − 114 ± 4.4031e − 114	8.2698e − 110 ± 0.0000e − 000	**1.6994e − 194 ± 0.0000e − 000**	6.6159e − 108 ± 0.0000e − 000
f_2	1.9281e − 007 ± 8.9376e − 007	7.4820e − 007 ± 3.0968e − 007	2.6862e − 007 ± 6.0065e − 007	**7.4820e − 008 ± 3.0968e − 008**
f_3	4.4415e − 016 ± 2.5108e − 016	8.9382e − 016 ± 8.1571e − 016	**8.8817e − 016 ± 0.0000e − 000**	2.1438e − 015 ± 2.6596e − 015
f_4	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	3.2039e − 015 ± 1.8147e − 015	**2.6532e − 015 ± 1.3037e − 015**	3.9092e − 015 ± 1.9459e − 015	4.3241e − 015 ± 2.9542e − 015
f_{10}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{11}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{12}	5.1900e + 001 ± 4.3580e + 001	4.1625e + 001 ± 9.1218e + 000	**2.6201e + 001 ± 8.3454e + 000**	2.9186e + 001 ± 6.7990e + 001
f_{13}	6.5345e + 001 ± 3.4082e + 001	5.1253e + 001 ± 2.3126e + 001	**4.5502e + 001 ± 1.0609e + 001**	4.8247e + 001 ± 1.0493e + 001
f_{14}	3.0358e − 003 ± 3.8244e − 003	4.2769e − 003 ± 2.5513e − 003	**2.4186e − 003 ± 5.3084e − 003**	3.4354e − 003 ± 4.1538e − 003
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	9.4154e − 002 ± 7.1725e − 002	8.8757e − 002 ± 1.4384e − 002	**8.8501e − 002 ± 1.5560e − 001**	9.6583e − 002 ± 1.7236e − 001

Table 4: Results (mean ± std) of RHCSA with varying recombination rate in 16 benchmark functions with $D = 10$.

Function	0.1	0.5	0.7	1
f_1	3.8119$e-119 \pm 3.8510e-119$	4.0163$e-178 \pm 0.0000e-000$	**7.2585e − 195 ± 0.0000e − 000**	3.0122$e-178 \pm 0.0000e-000$
f_2	2.9186$e-004 \pm 1.1966e-004$	3.1115$e-004 \pm 1.05044e-004$	8.9516$e-004 \pm 3.20393e-004$	**2.2544e − 004 ± 1.5452e − 004**
f_3	1.9741$e-015 \pm 1.9767e-015$	1.4807$e-015 \pm 1.9052e-015$	**8.8817e − 016 ± 0.0000e − 000**	1.4135$e-015 \pm 1.9876e-015$
f_4	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	5.6305$e-015 \pm 1.3377e-015$	7.9474$e-015 \pm 2.5291e-015$	**3.9292e − 015 ± 1.1420e − 015**	4.1535$e-015 \pm 1.2103e-015$
f_{10}	3.7179$e-002 \pm 1.8338e-002$	**3.1530e − 002 ± 1.7649e − 002**	3.8363$e-002 \pm 3.77691e-002$	3.7971$e-002 \pm 3.7825e-002$
f_{11}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{12}	2.3153$e+000 \pm 1.3464e+000$	2.3031$e+000 \pm 1.3465e+000$	1.5954$e+000 \pm 1.5954e+000$	**1.3685e + 000 ± 1.4679e + 000**
f_{13}	4.0817$e+000 \pm 5.5791e+000$	**3.8531e + 000 ± 2.2514e + 000**	4.0000$e+000 \pm 2.1213e+000$	4.8073$e+000 \pm 3.4243e+000$
f_{14}	**5.5773e − 008 ± 3.6818e − 007**	6.9679$e-008 \pm 5.9121e-007$	9.8137$e-008 \pm 7.7851e-007$	7.0125$e-008 \pm 1.0013e-007$
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	8.6735$e+000 \pm 8.3012e+000$	6.2013$e+000 \pm 4.2016e+000$	**5.2613e + 000 ± 5.0573e + 000**	5.6012$e+000 \pm 7.1232e+000$

Table 5: Results (mean ± std) of RHCSA with varying recombination rate in 16 benchmark functions with $D = 30$.

Function	0.1	0.5	0.7	1
f_1	1.6848$e-121 \pm 0.0000e-000$	1.1232$e-150 \pm 0.0000e-000$	1.6994$e-194 \pm 0.0000e-000$	**8.4243e − 198 ± 0.0000e − 000**
f_2	3.3269$e-007 \pm 8.1537e-007$	3.3493$e-007 \pm 3.1302e-007$	**2.6862e − 007 ± 6.0065e − 007**	7.3295$e-007 \pm 3.0183e-007$
f_3	9.6645$e-016 \pm 2.5121e-016$	8.9312$e-016 \pm 2.8421e-016$	**8.8817e − 016 ± 0.0000e − 000**	2.6645$e-015 \pm 2.7580e-015$
f_4	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	2.5600$e-015 \pm 2.2178e-015$	**2.4424e − 015 ± 1.2352e − 015**	3.9092$e-015 \pm 1.9459e-015$	2.5461$e-015 \pm 2.2735e-015$
f_{10}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{11}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{12}	4.6654$e+001 \pm 4.1276e+001$	2.7521$e+001 \pm 1.0387e+000$	**2.6201e + 001 ± 8.3454e + 000**	2.8012$e+001 \pm 1.2293e+001$
f_{13}	5.1231$e+001 \pm 1.3024e+001$	4.5613$e+001 \pm 1.3450e+001$	**4.5502e + 001 ± 1.0609e + 001**	4.7234$e+001 \pm 1.1249e+001$
f_{14}	2.6782$e-003 \pm 3.2933e-003$	2.8334$e-003 \pm 6.2013e-003$	**2.4186e − 003 ± 5.3084e − 003**	3.0187$e-003 \pm 8.7612e-003$
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	8.7011$e-002 \pm 5.6745e-002$	8.7329$e-002 \pm 9.2581e-002$	8.8501$e-002 \pm 1.5560e-001$	**6.0112e − 002 ± 2.5832e − 002**

Table 6: Results of traditional clonal selection algorithm and RHCSA when $D = 10$.

ALGs	Clonal selection algorithm (CSA)		CSA with recombination		RHCSA	
	Mean	Std	Mean	Std	Mean	Std
f_1	5.0414$e-008$	4.4098$e-008$	6.9076$e-056$	9.5731$e-055$	**7.2585e−195**	0
f_2	5.5057$e+000$	2.7914$e+000$	4.5314$e-003$	4.7750$e-002$	**8.9516e−004**	3.2039$e-004$
f_3	2.8532$e-003$	8.5712$e-003$	2.3976$e-015$	5.7314$e-015$	**8.8817e−016**	0
f_4	2.6092$e-002$	1.9092$e-002$	0	0	**0**	0
f_5	1.2569$e-002$	4.4216$e-003$	0	0	**0**	0
f_6	8.4141$e+000$	2.4848$e+000$	0	0	**0**	0
f_7	5.7681$e+000$	1.3571$e+000$	0	0	**0**	0
f_8	3.6436$e+002$	9.7648$e+001$	0	0	**0**	0
f_9	1.0619$e+000$	7.9728$e-001$	1.3592$e-008$	1.0975$e-008$	**3.9292e−015**	1.1420$e-015$
f_{10}	4.2974$e-001$	1.0264$e-001$	2.5301$e-001$	1.1356$e-001$	**1.6937e−002**	1.6091$e-002$
f_{11}	6.0837$e+000$	1.5496$e+000$	5.2403$e-008$	1.2560$e-008$	**0**	0
f_{12}	4.2193$e+001$	5.4173$e+000$	3.6021$e+001$	1.2541$e+001$	**2.7006e+000**	1.5954$e+000$
f_{13}	4.3672$e+001$	5.5570$e+000$	2.3702$e+001$	2.5091$e+001$	**4.0000e+000**	2.1213$e+000$
f_{14}	1.9305$e+003$	2.3358$e+002$	2.7201$e-003$	6.1034$e-003$	**9.8137e−008**	7.7851$e-007$
f_{15}	4.5354$e+001$	3.2450$e+001$	1.0276$e-033$	1.8734$e-033$	**0**	0
f_{16}	5.1701$e+001$	1.7347$e+001$	5.2613$e+000$	5.0573$e+000$	**5.2613e+000**	5.0573$e+000$

Table 7: Results of traditional clonal selection algorithm and RHCSA when $D = 30$.

ALGs	Clonal selection algorithm (CSA)		CSA with recombination		RHCSA	
	Mean	Std	Mean	Std	Mean	Std
f_1	6.4595$e-002$	2.9504$e-002$	1.4668$e-105$	3.0015$e-105$	**1.6994e−194**	0
f_2	2.7748$e+001$	2.1866$e+000$	1.8524$e-001$	2.6531$e-001$	**2.6862e−007**	6.0065$e-007$
f_3	3.1589$e-001$	1.8716$e-001$	**1.9187e−017**	3.8302$e-017$	8.8817$e-016$	0
f_4	1.6746$e-001$	4.3211$e-002$	0	0	**0**	0
f_5	3.6752$e+001$	4.6907$e+000$	0	0	**0**	0
f_6	3.6752$e+001$	4.6907$e+000$	0	0	**0**	0
f_7	2.3935$e+001$	2.5226$e+000$	0	0	**0**	0
f_8	1.6279$e+003$	1.7813$e+002$	0	0	**0**	0
f_9	3.8700$e+000$	3.2556$e-001$	1.8245$e-005$	1.3675$e-005$	**3.9092e−015**	1.9459$e-015$
f_{10}	8.3333$e-001$	7.5256$e-002$	2.9176$e-028$	1.3613$e-028$	**0**	0
f_{11}	3.3702$e+001$	2.7162$e+000$	1.1680$e-030$	1.9130$e-030$	**0**	0
f_{12}	2.5929$e+002$	1.4658$e+001$	4.3898$e+001$	2.0666$e+000$	**2.6201e+001**	8.3454$e+000$
f_{13}	2.5762$e+002$	2.1376$e+001$	5.4715$e+001$	1.7261$e+001$	**4.5502e+001**	1.0609$e+001$
f_{14}	8.7875$e+003$	3.2136$e+002$	2.5296$e-002$	1.6225$e-002$	**2.4186e−003**	5.3084$e-003$
f_{15}	4.4892$e+001$	9.5942$e+000$	3.3097$e-033$	1.6468$e-033$	**0**	0
f_{16}	3.9889$e+001$	4.3023$e+000$	2.5177$e-001$	1.1168$e-001$	**8.8501e−002**	1.5560$e-001$

real-coded genetic algorithm (GL-25) [49]; and comprehensive learning particle swarm optimization (CLPSO) [1].

Table 8 presents the mean values and standard deviations of the seven algorithms on the 16 test functions with $D = 10$, where the best results are shown in bold face. First of all, RHCSA performs the best on the 6 unrotated multimodal functions f_3~f_8. It could exactly locate the global optima in 7 functions, f_4~f_8, f_{11}, and f_{15}. In particular, it is observed that RHCSA surpasses all the other algorithms on functions f_3 and f_{12} with great superiority. Moreover, RHCSA, HLCSA, BCSA, SaDE, and CLPSO can obtain the global minima 0 on functions f_5, f_6, f_7, and f_8; RHCSA, HLCSA, BCSA, SaDE, and OXDE get the global optimum of function f_{11} and RHCSA, HLCSA, SaDE, OXDE, and GL-25 find the optimum on function f_{15}. Though RHCSA cannot get the best results on functions f_1, f_9, f_{10}, f_{13}, and f_{16}, it still gets the comparable solutions with respect to the best results obtained by the other algorithms. To the composite problems f_{15}, RHCSA reaches the global best as HLCSA, OXDE, SaDE, and GL-25. To the function f_{16}, RHCSA surpasses the other algorithms except for HLCSA.

Similar results can be observed from Table 9 when D equals 30. RHCSA can consistently obtain good results which are even slightly better than those on 10-D problems. RHCSA

Table 8: Results (mean ± std) of RHCSA and other state-of-the-art evolutionary algorithms when $D = 10$.

(a)

ALGs	f_1	f_2	f_3	f_4
RHCSA	7.2585$e-195 \pm 0.000e-000$	8.9516$e-004 \pm 3.2039e-004$	**8.8817e − 016 ± 0.000e − 000**	**0 ± 0**
HLCSA	4.2228$e-053 \pm 9.9892e-053$	**3.9087e − 028 ± 6.6496e − 028**	2.5757$e-015 \pm 4.8648e-016$	**0 ± 0**
BCSA	1.1694$e-037 \pm 1.3065e-037$	1.8521$e-001 \pm 7.2808e-001$	2.6645$e-015 \pm 0.0000e-000$	1.4146$e-001 \pm 1.3744e-001$
OXDE	4.5059$e-056 \pm 7.4966e-056$	1.0265$e-026 \pm 2.0786e-026$	2.0724$e-015 \pm 1.3467e-015$	9.9330$e-004 \pm 1.0673e-002$
SaDE	1.4451$e-176 \pm 0.0000e-000$	2.0249$e+000 \pm 7.4832e-001$	5.0330$e-015 \pm 1.1109e-015$	1.8074$e-003 \pm 3.8251e-003$
GL-25	**1.0771e − 321 ± 0.0000e − 000**	2.0956$e+000 \pm 6.3579e-001$	2.7830$e-015 \pm 1.4703e-015$	1.2134$e-002 \pm 1.0199e-002$
CLPSO	1.8154$e-041 \pm 3.0360e-041$	2.1490$e+000 \pm 1.2450e+000$	3.9672$e-015 \pm 1.7413e-015$	7.4577$e-006 \pm 2.1864e-005$

(b)

ALGs	f_5	f_6	f_7	f_8
RHCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
HLCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
BCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
OXDE	**0 ± 0**	6.6331$e-002 \pm 2.5243e-001$	5.6667$e-001 \pm 7.2793e-001$	**0 ± 0**
SaDE	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
GL-25	7.3315$e-007 \pm 2.2405e-006$	1.9633$e+000 \pm 1.1774e+000$	5.6336$e+000 \pm 1.2724e+000$	2.8952$e+002 \pm 1.9959e+002$
CLPSO	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**

(c)

ALGs	f_9	f_{10}	f_{11}	f_{12}
RHCSA	3.9292$e-015 \pm 1.1420e-015$	1.6937$e-002 \pm 1.6091e-002$	**0 ± 0**	**1.5954e + 000 ± 1.5954e + 000**
HLCSA	3.5527$e-015 \pm 0.0000e-000$	2.8802$e-002 \pm 1.9129e-002$	**0 ± 0**	4.2783$e+000 \pm 2.0095e+000$
BCSA	3.5527$e-015 \pm 0.0000e-000$	3.2715$e-001 \pm 1.7104e-001$	**0 ± 0**	6.2881$e+001 \pm 1.6334e+001$
OXDE	3.1974$e-015 \pm 1.0840e-015$	4.9045$e-001 \pm 2.7411e-002$	**0 ± 0**	3.7808$e+000 \pm 1.9094e+000$
SaDE	**9.4739e − 016 ± 1.5979e − 015**	1.3704$e-002 \pm 1.6048e-002$	**0 ± 0**	3.9135$e+000 \pm 1.4295e+000$
GL-25	3.5527$e-015 \pm 0.0000e-000$	**1.0545e − 002 ± 1.0858e − 002**	2.1771$e-004 \pm 4.7010e-004$	3.3619$e+000 \pm 2.2861e+000$
CLPSO	3.8606$e-014 \pm 5.8665e-014$	2.8592$e-002 \pm 1.5307e-002$	1.4403$e-002 \pm 1.2235e-002$	4.1634$e+000 \pm 9.0362e-001$

(d)

ALGs	f_{13}	f_{14}	f_{15}	f_{16}
RHCSA	4.0000$e+000 \pm 2.1213e+000$	9.8137$e-008 \pm 7.7851e-007$	**0 ± 0**	5.2613$e+000 \pm 5.0573e+000$
HLCSA	4.2442$e+000 \pm 2.6469e+000$	**0 ± 0**	**0 ± 0**	**4.6177e − 001 ± 8.1247e − 001**
BCSA	6.5935$e+001 \pm 1.0618e+001$	2.6391$e+002 \pm 7.9682e+001$	4.3387$e-031 \pm 6.5797e-031$	8.7291$e+000 \pm 1.7669e+001$
OXDE	3.0956$e+000 \pm 1.1245e+000$	1.5792$e+001 \pm 6.7669e+001$	**0 ± 0**	1.0047$e+001 \pm 3.0498e+001$
SaDE	3.9534$e+000 \pm 1.9500e+000$	2.0681$e+002 \pm 8.6302e+001$	**0 ± 0**	1.8019$e+001 \pm 3.7308e+001$
GL-25	7.3657$e+000 \pm 2.2262e+000$	5.2254$e+002 \pm 1.7963e+002$	**0 ± 0**	9.0000$e+001 \pm 3.0513e+001$
CLPSO	**2.0254e + 000 ± 1.0621e + 000**	3.1281$e+002 \pm 1.5723e+002$	2.3104$e-002 \pm 6.5636e-002$	6.0233$e+000 \pm 4.0698e+000$

achieves the best performance on functions f_3, f_4, f_5, f_6, f_7, f_8, f_{10}, f_{11}, f_{15}, and f_{16} compared with the other algorithms. Even RHCSA cannot find the best results on functions f_9, f_{12}, and f_{13}; its obtained results are very close to the best results obtained by the other algorithms. For example, the best result on function f_{12} ($1.6549e + 001 \pm 4.4609e + 000$) is found by OXDE while a comparable result obtained by RHCSA is $2.6201e + 001 \pm 8.3454e + 000$. RHCSA greatly improves the results on functions f_3, f_{14}, and f_{16}. It obtains comparatively good results on functions f_9, f_{12}, and f_{13}. It can be observed that the scalability of RHCSA is pretty well when dealing with high-dimensional optimization problems.

5. Conclusion

This paper presents a new clonal selection based algorithm, in which combinatorial recombination and hypermutation are introduced to enhance the search ability. The proposed algorithm is tested on 16 commonly used benchmark functions with unimodal, multimodal, rotation, and composite characteristics. Firstly, the CSAs with and without recombination

Table 9: Results (mean ± std) of RHCSA and other state-of-the-art evolutionary algorithms when $D = 30$.

(a)

ALGs	f_1	f_2	f_3	f_4
RHCSA	1.6994e − 194 ± 0.0000e − 000	7.4820e − 001 ± 3.0968e − 001	**8.8817e − 016 ± 8.8817e − 016**	**0 ± 0**
HLCSA	7.1289e − 066 ± 2.0169e − 065	**1.1617e − 015 ± 5.5448e − 015**	2.6645e − 015 ± 0.0000e − 000	**0 ± 0**
BCSA	2.9665e − 025 ± 8.4182e − 025	1.9018e + 001 ± 2.6925e + 000	1.5614e − 013 ± 3.9345e − 013	**0 ± 0**
OXDE	4.8545e − 059 ± 1.2064e − 058	2.6577e − 001 ± 1.0114e + 000	2.6645e − 001 ± 0.0000e − 000	2.8730e − 003 ± 5.6727e − 003
SaDE	9.1236e − 150 ± 4.4538e − 149	2.1973e + 001 ± 1.0132e + 000	7.7383e − 001 ± 6.0009e − 001	1.1999e − 002 ± 1.9462e − 002
GL-25	**5.3539e − 228 ± 0.0000e − 000**	2.0832e + 001 ± 8.6842e − 001	8.4969e − 014 ± 1.7664e − 013	9.7959e − 015 ± 3.2264e − 014
CLPSO	1.9761e − 029 ± 1.5041e − 029	1.7605e + 001 ± 3.6364e + 000	1.8415e − 014 ± 3.0495e − 015	1.1102e − 016 ± 3.2467e − 016

(b)

ALGs	f_5	f_6	f_7	f_8
RHCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
HLCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
BCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
OXDE	1.6214e − 003 ± 6.5408e − 003	9.4189e − 000 ± 2.0859e − 000	1.5100e + 001 ± 2.9868e + 000	3.9479e + 000 ± 2.1624e + 001
SaDE	9.5195e − 002 ± 1.5798e − 001	8.6230e − 001 ± 8.9502e − 001	6.3333e − 001 ± 7.6489e − 001	3.9479e + 001 ± 6.4747e + 001
GL-25	7.1724 − e004 ± 4.8027e − 004	2.3030e + 001 ± 8.4952e + 000	3.9096e + 001 ± 2.2071e + 001	3.5030e + 003 ± 6.8004e + 000
CLPSO	**0 ± 0**	**0 ± 0**	8.7634e − 015 ± 1.3333e − 014	**0 ± 0**

(c)

ALGs	f_9	f_{10}	f_{11}	f_{12}
RHCSA	3.9092e − 015 ± 1.9459e − 015	**0 ± 0**	**0 ± 0**	2.6201e + 001 ± 8.3454e + 000
HLCSA	**3.5527e − 015 ± 0.0000e − 000**	**0 ± 0**	**0 ± 0**	2.5471e + 001 ± 6.7663e + 000
BCSA	1.3086e − 013 ± 3.4341e − 013	1.2273e − 002 ± 2.2389e − 002	1.8516e − 014 ± 4.8488e − 014	3.2115e + 001 ± 9.0135e + 000
OXDE	**3.5527e − 015 ± 0.0000e − 000**	1.5612e − 003 ± 3.2032e − 003	1.4210e − 001 ± 2.3765e − 001	**1.6549e + 001 ± 4.4609e + 000**
SaDE	1.1708e + 000 ± 6.5356e − 001	1.3096e − 002 ± 2.6787e − 002	2.0504e + 000 ± 9.0887e − 001	2.5050e + 001 ± 6.6415e + 000
GL-25	1.1416e − 013 ± 1.6841e − 013	4.2040e − 015 ± 5.5414e − 015	5.6795e − 003 ± 2.6720e − 003	2.9464e + 001 ± 2.2594e + 001
CLPSO	1.4501e − 007 ± 7.0645e − 007	4.4152e − 007 ± 7.4331e − 007	1.7977e + 000 ± 6.1941e − 001	4.6287e + 001 ± 5.7149e + 000

(d)

ALGs	f_{13}	f_{14}	f_{15}	f_{16}
RHCSA	4.5502e + 001 ± 1.0609e + 001	**2.4186e − 003 ± 5.3084e − 003**	**0 ± 0**	**8.8501e − 002 ± 1.5560e − 001**
HLCSA	4.7609e + 001 ± 1.5104e + 001	1.0663e + 003 ± 5.2886e + 002	**0 ± 0**	3.2212e + 000 ± 1.0133e + 000
BCSA	2.6912e + 001 ± 1.2152e + 001	2.6533e + 003 ± 5.3339e + 002	2.0259e − 023 ± 6.3626e − 023	1.8053e + 001 ± 1.8282e + 001
OXDE	**1.7959e + 001 ± 5.1559e + 000**	4.3428e + 001 ± 8.8341e + 001	3.3333e + 000 ± 1.8257e + 001	2.5992e + 000 ± 5.6332e − 001
SaDE	2.2788e + 001 ± 6.6879e + 000	2.4742e + 003 ± 5.9013e + 002	1.1833e − 031 ± 2.1659e − 031	1.0208e + 001 ± 1.7851e + 001
GL-25	9.6862e + 001 ± 4.0914e + 001	3.2335e + 003 ± 5.9871e + 002	2.7878e − 028 ± 1.1207e − 027	5.2187e + 001 ± 2.1654e + 001
CLPSO	4.0333e + 001 ± 7.5039e + 000	2.6321e + 003 ± 3.3553e + 002	8.2952e − 005 ± 3.3295e − 004	7.9983e + 000 ± 1.6728e + 000

operator is compared. The experimental results indicate that the proposed recombination operator is able to improve the performance of the CSA. And then the modified hypermutation makes further improvement. Finally, the proposed algorithm is compared with the state-of-the-art algorithms. The experimental results show the competitiveness of the RHCSA.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61403349, no. 61402422, no. 61501405,

no. 61401404, and no. 61302118), Science and Technology Research Key Project of Basic Research Projects in Education Department of Henan Province (no. 15A520033, no. 14B520066, and no. 17B510011), and Doctoral Foundation of Zhengzhou University of Light Industry (no. 2013BSJJ044) and in part by the Program for Science and Technology Innovation Talents in Universities of Henan Province, under grant no. 17HASTIT022.

References

[1] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[2] A. R. Jordehi, "Enhanced leader PSO (ELPSO): a new PSO variant for solving global optimisation problems," *Applied Soft Computing*, vol. 26, pp. 401–417, 2015.

[3] A. R. Jordehi, "Seeker optimisation (human group optimisation) algorithm with chaos," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 27, no. 6, pp. 753–762, 2015.

[4] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[5] A. R. Jordehi, "Brainstorm optimisation algorithm (BSOA): an efficient algorithm for finding optimal location and setting of FACTS devices in electric power systems," *International Journal of Electrical Power & Energy Systems*, vol. 69, pp. 48–57, 2015.

[6] A. A. Heidari, R. A. Abbaspour, and A. R. Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks," *Neural Computing & Applications*, 2015.

[7] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[8] E. Hart, C. McEwan, J. Timmis, and A. Hone, "Advances in artificial immune systems," *Evolutionary Intelligence*, vol. 4, no. 2, pp. 67–68, 2011.

[9] Y. Xu, Y. Jiang, and W. Deng, "An novel immune genetic algorithm and its application in WWTP," in *Proceedings of the Chinese Automation Congress (CAC '15)*, pp. 802–808, November 2015.

[10] R. Daoudi, K. Djemal, and A. Benyettou, "Improving cells recognition by local database categorization in Artificial Immune System algorithm," in *Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS '15)*, pp. 1–6, Douai, France, December 2015.

[11] Z. Lu, G. Pei, B. Liu, and Z. Liu, "Hardware implementation of negative selection algorithm for malware detection," in *Proceedings of the IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC '15)*, pp. 301–304, IEEE, Singapore, June 2015.

[12] M. Pavone, G. Narzisi, and G. Nicosia, "Clonal selection: an immunological algorithm for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 53, no. 4, pp. 769–808, 2012.

[13] R. Liu, L. Jiao, X. Zhang, and Y. Li, "Gene transposon based clone selection algorithm for automatic clustering," *Information Sciences*, vol. 204, no. 20, pp. 1–22, 2012.

[14] L. D. S. Batista, F. G. Guimarães, and J. A. Ramirez, "A distributed clonal selection algorithm for optimization in electromagnetics," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1598–1601, 2009.

[15] J. Zheng, Y. Chen, and W. Zhang, "A survey of artificial immune applications," *Artificial Intelligence Review*, vol. 34, no. 1, pp. 19–34, 2010.

[16] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, London, UK, 2002.

[17] N. Cruz-Cortés, "Handling constraints in global optimization using artificial immune systems: a survey," in *Constraint-Handling in Evolutionary Optimization*, pp. 237–262, Springer, Berlin, Germany, 2009.

[18] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[19] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1574–1587, 2011.

[20] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[21] V. Cutello, G. Nicosia, and M. Pavone, "Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator," in *Proceedings of the ACM Symposium on Applied Computing (SAC '06)*, pp. 950–954, ACM, Dijon, France, April 2006.

[22] T. Jansen and C. Zarges, "Analyzing different variants of immune inspired somatic contiguous hypermutations," *Theoretical Computer Science*, vol. 412, no. 6, pp. 517–533, 2011.

[23] N. Khilwani, A. Prakash, R. Shankar, and M. K. Tiwari, "Fast clonal algorithm," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 1, pp. 106–128, 2008.

[24] H. Lu and J. Yang, "An improved clonal selection algorithm for job shop scheduling," in *Proceedings of the International Symposium on Intelligent Ubiquitous Computing and Education (IUCE '09)*, pp. 34–37, Chengdu, China, May 2009.

[25] X. Liu, L. Shi, R. Chen, and H. Chen, "A novel clonal selection algorithm for global optimization problems," in *Proceedings of the International Conference on Information Engineering and Computer Science (ICIECS '09)*, pp. 1–4, Wuhan, China, December 2009.

[26] L. N. De Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 699–704, IEEE, Honolulu, Hawaii, USA, May 2002.

[27] Z. Li, Y. Zhang, and H.-Z. Tan, "IA-AIS: an improved adaptive artificial immune system applied to complex optimization problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 4692–4700, 2011.

[28] Z. Li, C. He, and H.-Z. Tan, "AINet-SL: artificial immune network with social learning and its application in FIR designing," *Applied Soft Computing Journal*, vol. 13, no. 12, pp. 4557–4569, 2013.

[29] Z. Li, J. Li, and J. Zhou, "An improved artificial immune network for multimodal function optimization," in *Proceedings of the 26th Chinese Control and Decision Conference (CCDC '14)*, pp. 766–771, IEEE, Changsha, China, June 2014.

[30] G. P. Coelho and F. J. V. Zuben, "A concentration-based artificial immune network for continuous optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[31] G. P. Coelho, F. O. De Franca, and F. J. V. Zuben, "A concentration-based artificial immune network for combinatorial

optimization," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 1242–1249, New Orleans, La, USA, June 2011.

[32] G. P. Coelho and F. J. V. Zuben, "A concentration-based artificial immune network for multi-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 343–357, 2011.

[33] A. R. Yildiz, "An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry," *Journal of Materials Processing Technology*, vol. 209, no. 6, pp. 2773–2780, 2009.

[34] M. Gong, L. Zhang, L. Jiao, and W. Ma, "Differential immune clonal selection algorithm," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS '07)*, pp. 666–669, Xiamen, China, December 2007.

[35] M. Gong, L. Jiao, F. Liu, and W. Ma, "Immune algorithm with orthogonal design based initialization, cloning, and selection for global optimization," *Knowledge & Information Systems*, vol. 25, no. 3, pp. 523–549, 2010.

[36] M. Gong, L. Jiao, H. Du, and L. Bo, "Multi-objective immune algorithm with non-dominated neighbor-based selection," *Evolutionary Computation*, vol. 16, no. 2, pp. 225–255, 2008.

[37] R. Shang, L. Jiao, F. Liu, and W. Ma, "A novel immune clonal algorithm for MO problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 35–50, 2012.

[38] M. Gong, L. Jiao, and L. Zhang, "Baldwinian learning in clonal selection algorithm for optimization," *Information Sciences*, vol. 180, no. 8, pp. 1218–1236, 2010.

[39] M. Gong, L. Jiao, J. Yang, and F. Liu, "Lamarckian learning in clonal selection algorithm for numerical optimization," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 1, pp. 19–37, 2010.

[40] Y. Peng and B. Lu, "Hybrid learning clonal selection algorithm," *Information Sciences*, vol. 296, pp. 128–146, 2015.

[41] L. N. De Castro and F. J. V. Zuben, "Artificial immune systems: part I—basic theory and applications," Tech. Rep. 210, Universidade Estadual de Campinas, 1999.

[42] H. Bersini, "The immune and the chemical crossover," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 306–313, 2002.

[43] Y. Qi, Z. Hou, M. Yin, H. Sun, and J. Huang, "An immune multi-objective optimization algorithm with differential evolution inspired recombination," *Applied Soft Computing Journal*, vol. 29, pp. 395–410, 2015.

[44] R. Liu, C. L. Ma, F. He, W. P. Ma, and L. C. Jiao, "Reference direction based immune clone algorithm for many-objective optimization," *Frontiers in Computer Science*, vol. 8, no. 4, pp. 642–655, 2014.

[45] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1011–1024, 2013.

[46] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 68–75, Pasadena, Calif, USA, June 2005.

[47] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.

[48] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[49] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1088–1113, 2008.