Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 128902, 15 pages
http://dx.doi.org/10.1155/2015/128902

Research Article

The Cellular Differential Evolution Based on Chaotic Local Search

Qingfeng Ding[1,2] and Guoxin Zheng[1]

[1]*Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200072, China*
[2]*School of Electrical and Electronic Engineering, East China Jiaotong University, Nanchang 330013, China*

Correspondence should be addressed to Qingfeng Ding; brandy724@sina.com

Received 27 January 2015; Revised 30 April 2015; Accepted 4 May 2015

Academic Editor: George S. Dulikravich

To avoid immature convergence and tune the selection pressure in the differential evolution (DE) algorithm, a new differential evolution algorithm based on cellular automata and chaotic local search (CLS) or ccDE is proposed. To balance the exploration and exploitation tradeoff of differential evolution, the interaction among individuals is limited in cellular neighbors instead of controlling parameters in the canonical DE. To improve the optimizing performance of DE, the CLS helps by exploring a large region to avoid immature convergence in the early evolutionary stage and exploiting a small region to refine the final solutions in the later evolutionary stage. What is more, to improve the convergence characteristics and maintain the population diversity, the binomial crossover operator in the canonical DE may be instead by the orthogonal crossover operator without crossover rate. The performance of ccDE is widely evaluated on a set of 14 bound constrained numerical optimization problems compared with the canonical DE and several DE variants. The simulation results show that ccDE has better performances in terms of convergence rate and solution accuracy than other optimizers.

1. Introduction

Differential evolution (DE) algorithm, proposed by Storn and Price [1], is a population-based parallel iterative optimization algorithm and outperforms many other optimization methods in terms of convergence speed and robustness over common benchmark functions and real-world problems [2]. Its optimization performance is mainly influenced by three critical control parameters including scaling factor F, population size NP, and crossover rate CR. Nevertheless, DE algorithm also has immature convergence and search stagnation and other defects, which limit its application range and its ability of optimization, which urgently need to be explored in depth.

Most evolutionary algorithms use a single population of individuals and apply stochastic operators to them as a whole [3]. Some recent studies show that it is easy to implement the parallel computing for those evolutionary algorithms with spatial structure [4]. However, the canonical DE algorithm ignores the spatial structure and the complex interaction among individuals of local groups during evolutionary processes. For example, the population of the canonical DE will not be affected by external disturbances and the individual states. Namely, the death, resurrection, and migration of the individual will never be modified, which is obviously inconsistent with the actual process of biological evolution. Among numerous evolutionary algorithms with spatial structure [5], the cellular evolutionary algorithm (cEA) is a kind of evolutionary algorithms of discrete groups based on spatial structure. It means an individual may interact with its adjacent neighbors. The overlapped neighborhoods of cEA help in exploring the search space while exploitation takes place inside neighborhood by stochastic operators. Dorronsoro and Bouvry [6] proposed a new cellular genetic algorithm (cGA) that automatically manages its neighbors according to the quality of individuals in a population and sets the most reasonable population structure based on the convergence speed. Then the traditional parameters configuration for the population and the neighbor structure becomes less necessary. According to the evolution characteristics of CA in

a discrete space and ant optimization, a CA-based optimization method is proposed for solving optimization problems with geometric constraints [7]. Lu et al. [8] proposed a cellular GA with the evolutionary rules and derived the selection of the evolutionary rules, and this GA variant has a better ability to maintain the diversity of the population relative to canonical genetic algorithm. Lorenzo and Glisic [9] presented a novel sequential genetic algorithm (SGA) to optimize the relaying topology in multihop cellular networks aware of the intercell interference and the spatial traffic distribution dynamics. Noman and Iba [10] proposed cellular differential evolution (cDE) algorithm with linear and compact neighbor, which only uses the cellular neighbor but does not consider the dynamic population evolutionary. Noroozi et al. [11] proposed CellularDE to address dynamic optimization problems, which limits the number of individuals in each cell to prevent convergence and maintain diversity of the population. Jia et al. [12] proposed that chaotic local search (CLS) can enhance the performance of DE.

To simulate the natural condition, the paper proposes a CLS-based cDE (ccDE) algorithm to simulate its dynamic process. First of all, the algorithm employs CA with the parallel evolution to balance the exploration and exploitation tradeoff for DE. Next, the CLS helps to avoid immature convergence and search stagnation by utilizing chaotic search with the ergodicity and randomicity. At last, the ccDE employs the orthogonal crossover operator to replace the binomial crossover operator in canonical DE, which contributes to the improvement of the convergence speed and maintenance of the population diversity.

The remainder of this paper is organized as follows. Sections 2 and 3 describe the canonical DE and the ccDE, respectively. Section 4 briefly illustrates the comparison of the performance of the proposed ccDE algorithm with the canonical DE algorithm and four DE variants over a suit of 14 problems. The findings of the paper are demonstrated in Section 5.

2. Canonical DE

DE makes use of a new mutation operator which depends on the differences among randomly selected pairs of individuals instead of predetermined probability distribution function [1]. Its whole evolutionary process consists of two stages: initialization stage and iterative evolutionary stage.

In the initialization stage, the initial individual population is chosen at random in a D dimensional search space. A differential evolutionary population of gth generation is as follows:

$$\begin{aligned} \mathbf{P}^g(\mathbf{X}) &= \left[\mathbf{X}_1^g, \ldots, \mathbf{X}_i^g, \ldots, \mathbf{X}_{NP}^g\right], \quad i = 0, 1, \ldots, NP, \\ \mathbf{X}_i^g &= \left[x_{i,1}^g, x_{i,2}^g, \ldots, x_{i,j}^g, \ldots, x_{i,D}^g\right], \\ & \qquad j = 1, 2, \ldots, D, \end{aligned} \tag{1}$$

wherein $\mathbf{X}_i^g$ denotes ith individual of gth generation which denotes the generation counter, NP denotes the population size, D denotes the dimension of decision space, and $\mathbf{P}^0(\mathbf{X})$ denotes the initial population.

In the iterative evolutionary stage, each individual goes through a succession of iterative processes including mutation, crossover, and selection. Then the individual is evaluated and updated by utilizing the fitness function. The above iterative evolutionary stage is repeated generation after generation until the termination criterion is met.

(1) Mutation. The mutant individual of DE is decided from the differences among randomly selected pairs of individuals. For each target individual $\mathbf{X}_i^g$, a mutant individual is generated as follows:

$$\mathbf{V}_i^g = \mathbf{X}_{p_1}^g + F \cdot \left(\mathbf{X}_{p_2}^g - \mathbf{X}_{p_3}^g\right), \tag{2}$$

wherein $\mathbf{X}_{p_1}^g$, $\mathbf{X}_{p_2}^g$, and $\mathbf{X}_{p_3}^g$ are the best individual or randomly selected individuals from the current population, all of which are different from the target individual $\mathbf{X}_i^g$. The scaling factor F is a predefined constant for scaling the differential individual vector in (2).

(2) Crossover. In the canonical version, DE applies binomial crossover operator to generate a trial individual by recombining a target individual and its corresponding mutant individual. The binomial crossover can be outlined as follows:

$$u_{ij}^g = \begin{cases} v_{ij}^g, & \text{rand}(0,1) \le \text{CR} \\ x_{ij}^g, & \text{otherwise,} \end{cases} \tag{3}$$

wherein u_{ij}^g, v_{ij}^g, and x_{ij}^g are an element of the trial individual $\mathbf{U}_i^g$, the mutant individual $\mathbf{V}_i^g$, and the target individual $\mathbf{X}_i^g$, respectively. The crossover factor $\text{CR} \in (0, 1)$ is a predefined constant to adapt to different optimization domain, which is called crossover probability.

(3) Selection. The selection scheme in DE adopts one-to-one competitive strategy. The trial individual $\mathbf{U}_i^g$ competes with the target individual $\mathbf{X}_i^g$, which is decided by the fitness value of their individuals. The selection scheme can be outlined as follows:

$$\mathbf{X}_i^{g+1} = \begin{cases} \mathbf{U}_i^g, & \text{if } f\left(\mathbf{U}_i^g\right) \le f\left(\mathbf{X}_i^g\right) \\ \mathbf{X}_i^g, & \text{otherwise.} \end{cases} \tag{4}$$

3. ccDE Algorithm

3.1. The Evolution Mechanism of Cellular Automata. CA proposed by Neumann [13] is a highly parallel computing model. For the most usual 2D CA, every cell is arranged into a 2D toroidal mesh and follows the same rule and updates on the basis of the local rule synchronously. Different evolutionary rules produce different cellular states. The basic model of the evolution rule is defined as follows:

$$\text{if } S_t = 1,$$

$$\text{then } S_{t+1} = \begin{cases} 1, & S \in S1 \\ 0, & S \notin S1; \end{cases}$$

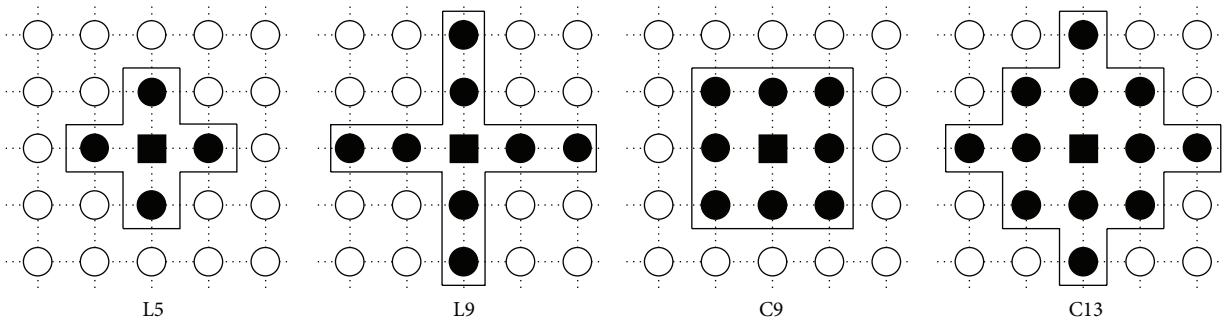


Figure 1: Typical neighborhood structures in cEAs.

$$\text{if } S_t = 0,$$
$$\text{then } S_{t+1} = \begin{cases} 1, & S \in S2 \\ 0, & S \notin S2, \end{cases} \tag{5}$$

wherein S_t and S_{t+1} denote the cellular state of tth step and $(t+1)$th step. $S1$ is a set of the numbers of active neighbors needed for the active cell to stay alive and $S2$ is a set of the numbers of active neighbors in order to resuscitate a dead cell.

The neighborhood structure actually affects the quality of the search, such as the improvement of inferior solutions, the avoidance of local optimum, and the maintenance of the population diversity. The two most common cellular neighbor structures are linear (L) pattern and compact (C) pattern. Figure 1 shows four typical neighborhood structures employed in cEAs.

3.2. Chaotic Local Search. Because of the ergodicity and randomicity, a chaotic system changes randomly, but it eventually goes through every state with a long time duration. The characteristics of chaotic systems can be used to build up a search operator to optimize its objective functions. Probably, the local search may lead to unacceptably rapid convergence; hence it will result in a problem that the whole evolutionary process would trap into local optima when exploring a large search space. To avoid this problem, a CLS with shrinking strategy is utilized with the optimization process [12].

In this paper, the Logistic chaotic function is employed as follows:

$$\beta_j^{k+1} = \mu \beta_j^k \left(1 - \beta_j^k\right), \quad k = 1, 2, \ldots, \tag{6}$$

where β_j^k is the jth chaotic variable in k generation and the Logistic function comes into a thorough chaotic state when $\mu = 4$. The formula of chaotic local search is

$$\mathbf{X}_i^{g(\text{new})} = (1 - \lambda)\, \mathbf{X}_i^g + \lambda \beta, \tag{7}$$

where $\mathbf{X}_i^{g(\text{new})}$ is a new individual vector of $\mathbf{X}_i^g$ in gth generation generated by chaotic local search and β is generated by the Logistic chaotic function. Meanwhile, λ is the shrinking scale related to the current function evaluations (FEs) and m decides the shrinking speed while more accurate results can be obtained on most functions when $m = 1500$ [12]:

$$\lambda = 1 - \left| \frac{\text{FEs} - 1}{\text{FEs}} \right|^m. \tag{8}$$

3.3. Orthogonal Crossover. Mostly, we did not have the prior knowledge about the optimization problem [14], so it may be very hard to choose an appropriate crossover strategy and crossover rate. To seek the best combination of the mutant and target individual for the trial individual, it is necessary for a scientific experiment to select the best one from all possible combinations. However, it is not possible to test all combinations for cost effectiveness. It is advisable to select a small but representative sample for testing [15].

For the above purpose, $L_Q(n^D)$ is employed as the orthogonal array for D factors, n levels, and Q combinations of levels. There are Q rows for this array in which every row is a combination of levels. For $L_Q(2^D)$ in this paper, 2 represents that all factors have two levels: 0 and 1. In like manner, it is a similar process to choose every dimension parameter of a trial individual from the target individual or its matching individual of DE. D is the dimension of the individual vector of DE, and Q is the total number of experiments. The design of the orthogonal embedded crossover operation is called orthogonal crossover.

3.4. Algorithmic Description of ccDE. The sizes of individual vectors and cells are both $NP = np \times np$, while the value of a cell means live or dead status of a vector. The proposed ccDE algorithm is elucidated as follows.

Step 1. Initialization is the first step.

Step 1.1. Set the population size NP, the generation counter $G = 0$, the initialization value of the scaling factor F, and the termination criteria of the algorithm.

Step 1.2. Initialize $NP = np \times np$ individuals composing the initial population in the parameter space:

$$\mathbf{P}^G = \begin{bmatrix} \mathbf{X}_{1,1}^G & \cdots & \mathbf{X}_{1,j}^G & \cdots & \mathbf{X}_{1,np}^G \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{X}_{i,1}^G & \cdots & \mathbf{X}_{i,j}^G & \cdots & \mathbf{X}_{i,np}^G \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{X}_{np,1}^G & \cdots & \mathbf{X}_{np,i}^G & \cdots & \mathbf{X}_{np,np}^G \end{bmatrix}_{np \times np}, \quad (9)$$

with $\mathbf{X}_{i,j}^G = \{x_{i,j}^{G,1}, x_{i,j}^{G,2}, \ldots, x_{i,j}^{G,D}\}$ uniformly distributed in the range $[\mathbf{X}_l, \mathbf{X}_u]$, where $\mathbf{X}_l = \{x_l^1, \ldots, x_l^D\}$ and $\mathbf{X}_u = \{x_u^1, \ldots, x_u^D\}$.

Step 1.3. Initialize the cellular automata with $NP = np \times np$ torus topology which mean the living condition of $NP = np \times np$ individual vectors:

$$\mathbf{C}^G = \begin{bmatrix} c_{1,1} & \cdots & c_{1,j} & \cdots & c_{1,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{i,1} & \cdots & c_{i,j} & \cdots & c_{i,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{np,1} & \cdots & c_{np,j} & \cdots & c_{np,np} \end{bmatrix}_{np \times np}, \quad (10)$$

with $c_{i,j} = \begin{cases} 0, & \text{rand}<0.5 \\ 1, & \text{otherwise} \end{cases}$, where 0 indicates the cell is dead and 1 indicates the cell is alive.

Step 1.4. Generate the orthogonal crossover table $L_Q(n^D)$.

Step 2. Target vectors $\mathbf{X}_{i,j}^G$ evolution for matching living cells with $c_{i,j} = 1$. If the cell matching current target vector is alive, jump to Step 2.1, or jump to Step 2. Only if live cells never exist, then jump to Step 3 and $G + 1$.

Step 2.1. Select several candidate vectors $\{\mathbf{X}_{p1}^G, \mathbf{X}_{p2}^G, \mathbf{X}_{p3}^G\}$ from the neighbors of current target vector $\mathbf{X}_{i,j}^G$ specific neighborhood model for mutation operation according to their fitness value:

$$\left\{\mathbf{X}_{p1}^G, \mathbf{X}_{p2}^G, \mathbf{X}_{p3}^G\right\} = \text{Select}\left(\text{Neighbors}\left(\mathbf{X}_{i,j}^G\right)\right). \quad (11)$$

Step 2.2. Execute the mutation operation to obtain the mutant individual according to (2) and select mutation strategy:

$$\mathbf{V}_{i,j}^G = \mathbf{X}_{p1}^G + F \cdot \left(\mathbf{X}_{p2}^G - \mathbf{X}_{p3}^G\right). \quad (12)$$

Step 2.3. Execute the orthogonal crossover operation to obtain the trial individual.

After getting the orthogonal crossover table $L_Q(2^D) = [\boldsymbol{\alpha}_1^T, \ldots, \boldsymbol{\alpha}_q^T, \ldots, \boldsymbol{\alpha}_Q^T]^T$, where $\boldsymbol{\alpha}_q = [\alpha_{q,1}, \ldots, \alpha_{q,d}, \ldots, \alpha_{q,D}]$, we apply those Q combinations to generate the following Q chromosomes $\widehat{\mathbf{U}}_{i,j}^G = [\mathbf{u}_1^T, \ldots, \mathbf{u}_q^T, \ldots, \mathbf{u}_Q^T]^T$ with $\mathbf{u}_q = [u_{q,1} \ \cdots \ u_{q,d} \ \cdots \ u_{q,D}]$, where

$$u_{q,d} = \begin{cases} x_{i,j}^{G,d}, & \text{if } \alpha_{q,d} = 1 \\ v_{i,j}^{G,d}, & \text{otherwise.} \end{cases} \quad (13)$$

Then select the best fitness from Q chromosomes, $\mathbf{U}_{i,j}^G = \arg\min_{\{\mathbf{u}_1, \ldots, \mathbf{u}_q, \ldots, \mathbf{u}_Q\}} \text{fitness}(\mathbf{u}_q)$.

Step 2.4. Evaluate the trial individual vector $\mathbf{U}_{i,j}^G$ and target individual vector $\mathbf{X}_{i,j}^G$ according to the fitness value then select the winner for the next generation according to (4).

Step 3. Execute the CLS on the best individual according to (7).

Step 4. Update each cell survival of new generation $\mathbf{C}^G$ on the basis of the evolutional rule of cellular automata according to (5).

Step 5. Output the result if the termination criterion is met; otherwise jump back to Step 2.

4. Experimental Results and Discussions

4.1. Description of Problems. To evaluate and compare the proposed ccDE algorithm with the canonical DE [1], SDE [16], jDE [17], SaDE [18], and DECLS [12], experiments are conducted on a set of 14 problems with different characteristics [18], including unimodal problems f_1–f_3 and multimodal problems f_4–f_{14}. Thereinto, f_1–f_4, f_6, f_8, f_{10}, and f_{11} are shifted for solving the problem that global optimum lies at the center of the search range. The problems f_5, f_7, f_9, and f_{12} are rotated to avoid local optimum lying along the coordinate axes or suffering from global optimum lying at the center of the search range. The problems f_{13} and f_{14} are built by utilizing some basic problems to obtain one more challenging problem. Table 1 lines the search range and global optimum of 14 problems.

4.2. Parameter Setting for Comparison. In this paper, experiments are conducted for 25 times independently of the above 14 problems. The control parameters F, CR, and NP of the canonical DE and four DE variants are set as in [17]. For the sake of fairness, the maximum number of FEs is configured on to be 100 000 for all problems. For the fixed dimension, the orthogonal crossovers of the 10D problems employ the tailor-made array $L_{12}(2^{10})$ and the first 10 columns of $L_{12}(2^{11})$, which is shown in Table 2.

In Table 2, 0 or 1 represents that one-dimension parameter of a trial individual is selected from the target individual or its matching individual of DE. The total number of experiments is 12. The orthogonal crossover of the 30D problem may be composed of three arrays of $L_{12}(2^{10})$ side by side.

Table 1: Search range and global optimum of 14 problems.

f	Problem_name	Dimension	Search range	Global optimum
1	sphere_func	10/30	[−100, 100]	0
2	schwefel_102	10/30	[−100, 100]	0
3	rosenbrock	10/30	[−100, 100]	0
4	ackley	10/30	[−32, 32]	0
5	ackley_rot	10/30	[−32, 32]	0
6	griewank	10/30	[−600, 600]	0
7	griewank_rot	10/30	[−600, 600]	0
8	rastrigin	10/30	[−5, 5]	0
9	rastrigin_rot	10/30	[−5, 5]	0
10	rastrigin_noncon	10/30	[−5, 5]	0
11	schwefel	10/30	[−500, 500]	0
12	schwefel_rot	10/30	[−500, 500]	0
13	com_func1	10/30	[−5, 5]	0
14	hybrid_func2	10/30	[−5, 5]	0

Table 2: The array of the first 10 columns of $L_{12}(2^{11})$.

Combinations	Level of each factor									
	1	2	3	4	5	6	7	8	9	10
1st	1	1	1	1	1	1	1	1	1	1
2nd	1	1	1	1	1	0	0	0	0	0
3rd	1	1	0	0	0	1	1	1	0	0
4th	1	0	1	0	0	1	0	0	1	1
5th	1	0	0	1	0	0	1	0	1	0
6th	1	0	0	0	1	0	0	1	0	1
7th	0	1	0	0	1	1	0	0	1	0
8th	0	1	0	1	0	0	0	1	1	1
9th	0	1	1	0	0	0	1	0	0	1
10th	0	0	0	1	1	1	1	0	0	1
11th	0	0	1	0	1	0	1	1	1	0
12th	0	0	1	1	0	1	0	1	0	0

The cellular evolution rule of the ccDE algorithm is $S1 = 1, 2, 3, 4$ and $S2 = 4, 5, 6, 7$ [9]. Its neighbor structure will use the optimum C9 neighborhood model [10].

4.3. Comparison of the Final Solutions Accuracy. In order to illustrate the outstanding accuracy of ccDE comparing with the canonical DE algorithm and four DE variants, the mean values and standard deviations of the best values are calculated by utilizing the results of the conducted 25 times independently of the 14 problems. Tables 3 and 4 show, respectively, the result of the mean values and standard deviations for the 10D and 30D problems, respectively, where the best value for each problem has been typed in bold.

From Tables 3 and 4, it can be seen that the ccDE algorithm outperforms the canonical DE and four DE variants. For low-dimension problems (10D), the ccDE algorithm obtains the smallest mean values, even theoretical optimum of all 14 problems. For the high-dimension problems (30D), the ccDE algorithm obtains smaller mean values of 13 problems, while the magnitude of each solution obtained by ccDE is equal to the optimal solution obtained by SDE over the f_{14} problem. Therefore the mean solution quality of ccDE is better and its solution quality is more stable than other DE variants.

4.4. The Wilcoxon Matched-Pairs Signed-Ranks Test. In order to illustrate the above statement, the Wilcoxon matched-pairs signed-ranks test is employed to compare ccDE with the canonical DE and four DE variants, respectively. Table 5 is the Wilcoxon p values of the mean data in Tables 3 and 4.

As can be seen from Table 5, the ccDE is more outstanding than canonical DE and four DE variants with the significance level $\alpha = 0.05$ considering independent matched-pairs

Table 3: The mean and standard deviations of optimum (10D).

Problems		DE	SaDE	SDE	jDE	DECLS	ccDE
f_1	Mean	$2.80E-17$	$8.77E-16$	$4.02E-19$	$1.69E-13$	$1.30E-20$	**0.00E + 00**
	Std	$9.27E-18$	$7.48E-16$	$2.82E-19$	$8.60E-14$	$3.49E-21$	$0.00E+00$
f_2	Mean	$1.96E+00$	$2.16E-13$	$2.29E-02$	$5.11E-04$	$4.75E-01$	**0.00E + 00**
	Std	$5.39E-01$	$1.13E-13$	$1.38E-02$	$3.45E-04$	$1.37E-01$	$0.00E+00$
f_3	Mean	$5.01E+00$	$2.30E+00$	$4.02E+00$	$5.42E+00$	$5.24E+00$	**1.41E + 00**
	Std	$1.51E+00$	$4.15E-01$	$2.25E+00$	$6.30E-01$	$1.12E+00$	$6.64E-01$
f_4	Mean	$3.56E-09$	$1.30E-08$	$2.59E-10$	$1.72E-07$	$6.21E-11$	**0.00E + 00**
	Std	$1.14E-09$	$1.50E-09$	$7.68E-11$	$2.76E-08$	$8.33E-12$	$0.00E+00$
f_5	Mean	$1.14E-08$	$1.42E-08$	$3.68E-10$	$4.41E-07$	$2.59E-10$	**3.55E − 15**
	Std	$4.40E-09$	$4.71E-09$	$3.66E-11$	$1.03E-07$	$5.28E-11$	$0.00E+00$
f_6	Mean	$1.01E-01$	$5.08E-02$	$1.04E-15$	$5.35E-03$	$5.69E-02$	**0.00E + 00**
	Std	$3.52E-02$	$1.83E-02$	$6.16E-16$	$2.35E-03$	$1.77E-02$	$0.00E+00$
f_7	Mean	$3.49E-01$	$9.00E-02$	**2.86E − 02**	$1.72E-01$	$3.91E-01$	$4.92E-02$
	Std	$8.42E-02$	$3.80E-02$	$5.58E-02$	$3.45E-02$	$3.15E-02$	$2.59E-02$
f_8	Mean	$1.34E-01$	$4.28E-01$	**0.00E + 00**	$5.83E-05$	$2.42E-05$	**0.00E + 00**
	Std	$1.45E-01$	$3.09E-01$	$0.00E+00$	$6.26E-05$	$1.84E-05$	$0.00E+00$
f_9	Mean	$2.47E+01$	$1.48E+01$	$1.75E+01$	$1.80E+01$	$1.32E+01$	**6.07E + 00**
	Std	$3.79E+00$	$2.38E+00$	$2.74E+00$	$2.34E+00$	$7.33E+00$	$1.86E+00$
f_{10}	Mean	$4.01E+00$	$1.39E+00$	**0.00E + 00**	$8.70E-02$	$1.97E+00$	**0.00E + 00**
	Std	$4.03E-01$	$5.69E-01$	$0.00E+00$	$7.80E-02$	$4.46E-01$	$0.00E+00$
f_{11}	Mean	$2.67E-10$	$1.71E-06$	**0.00E + 00**	$9.17E-06$	$1.82E-13$	**0.00E + 00**
	Std	$3.91E-10$	$9.74E-07$	$0.00E+00$	$7.15E-06$	$4.07E-13$	$0.00E+00$
f_{12}	Mean	$7.40E+02$	$4.44E+02$	$9.45E-09$	$2.74E-08$	$8.97E+02$	**0.00E + 00**
	Std	$3.18E+02$	$3.08E+02$	$1.31E-08$	$2.13E-08$	$2.68E+02$	$0.00E+00$
f_{13}	Mean	$2.17E-13$	$3.39E-16$	$2.75E-10$	$1.58E-12$	$2.71E-11$	**0.00E + 00**
	Std	$4.84E-13$	$1.85E-16$	$6.15E-10$	$2.13E-12$	$6.06E-11$	$0.00E+00$
f_{14}	Mean	$2.24E+00$	$1.95E-07$	$2.21E-10$	$3.15E-01$	$2.16E+00$	**0.00E + 00**
	Std	$8.77E-01$	$2.37E-07$	$4.70E-10$	$6.87E-01$	$1.26E+00$	$0.00E+00$

comparisons. The p values of multiple comparisons are as follows for 10D and 30D, respectively:

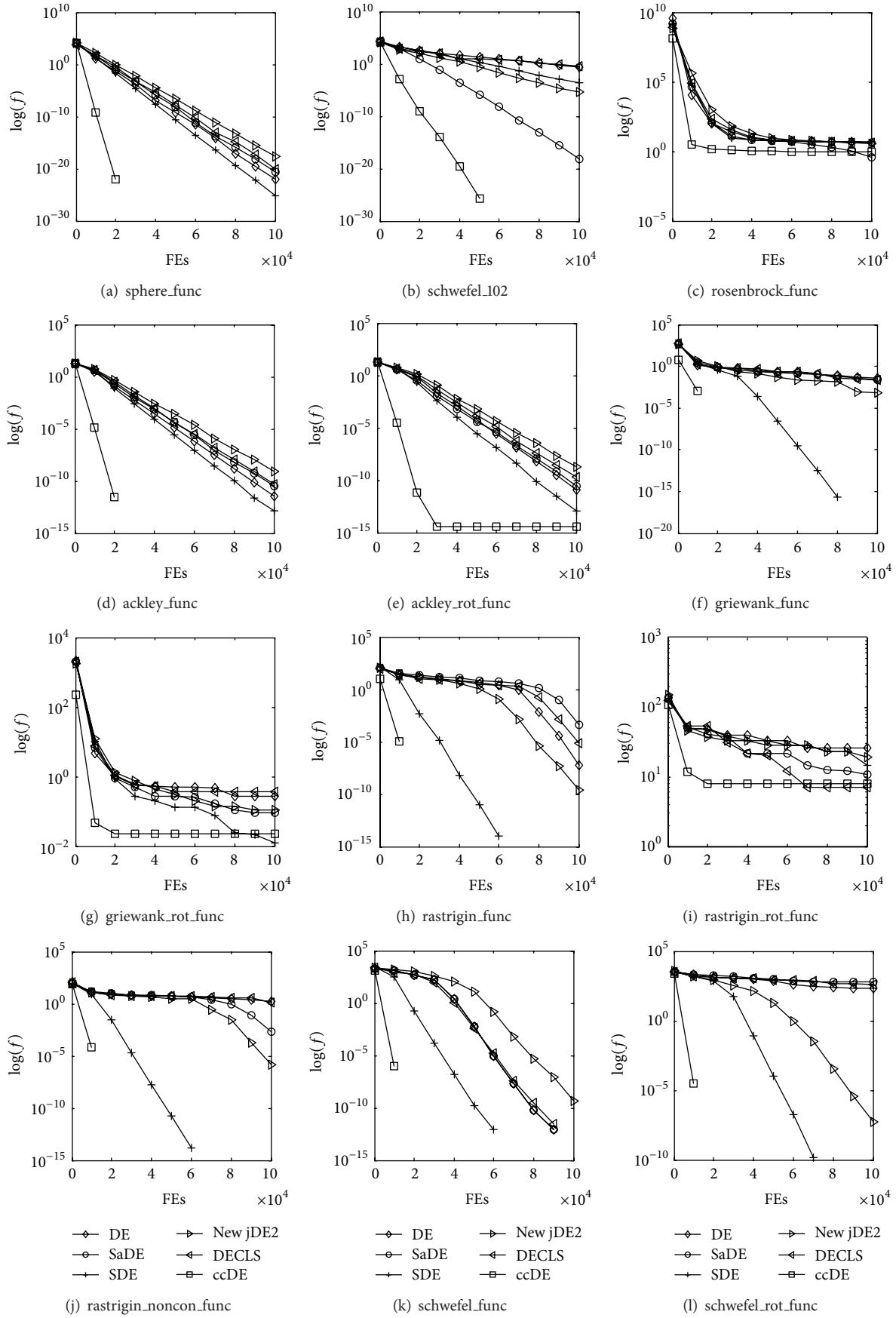
$$\begin{aligned} p_1 &= 1-(1-1.0E-3)(1-1.0E-3)(1-2.6E-2)\\ &\quad\cdot(1-1.0E-3)(1-1.0E-3)=3.98E-2,\\ p_2 &= 1-(1-1.0E-3)(1-1.0E-3)(1-8.0E-3)\\ &\quad\cdot(1-1.09E-1)(1-1.0E-3)=1.19E-2. \end{aligned} \tag{14}$$

As can be seen from the results of (14), the ccDE is superior to the canonical DE and four DE variants with p value of $p_1 = 3.98E-2$ for 10D and $p_2 = 1.19E-2$ for 30D. Furthermore, the results of the Wilcoxon matched-pairs signed-ranks test affirm that the ccDE is more outstanding for high-dimensional problems.

4.5. The Analysis of Convergence Characteristics. The convergence property of an evolutionary algorithm is a very important indicator of performance comparison. Figures 2 and 3 illustrate the convergence performance in terms of the median run of the best fitness value of the ccDE, the canonical DE, and four DE variants on 10D and 30D problems, respectively.

Initially, as can be observed in Figures 2 and 3 the ccDE obtains better adaptation values than the canonical DE and four DE variants on the initial stage. Furthermore, the convergence map between ccDE and other DE variants shows that the ccDE always converges faster and approaches the global optimum more easily than others on all 14 problems with 10D in Figure 2. It can be observed that the SDE converges fastest on f_{14}, followed by ccDE. What is more, for the 30D problems, all algorithms have great difficulty to find the global optimum on about half of the 14 problems, while the ccDE performs better on most of all problems in Figure 3.

In order to analyze the impact of cellular automata on the convergence performance, the DECLS algorithm is selected for the comparison. The difference between DECLS and ccDE is that the interaction among individuals in ccDE is limited in cellular neighbors while the individual state will evolve by the rule of the cellular evolution. Figure 4 illustrates the convergence properties in terms of the best fitness value between ccDE and DECLS on 10D and 30D problems f_1–f_{14}. From Figure 4, it can be observed that ccDE converges faster than DECLS on most of the 14 problems. Accordingly, the ccDE algorithm requires fewer FEs than the DECLS algorithm and therefore it has a lower time complexity.



(a) sphere_func (b) schwefel_102 (c) rosenbrock_func

(d) ackley_func (e) ackley_rot_func (f) griewank_func

(g) griewank_rot_func (h) rastrigin_func (i) rastrigin_rot_func

(j) rastrigin_noncon_func (k) schwefel_func (l) schwefel_rot_func

Figure 2: Continued.

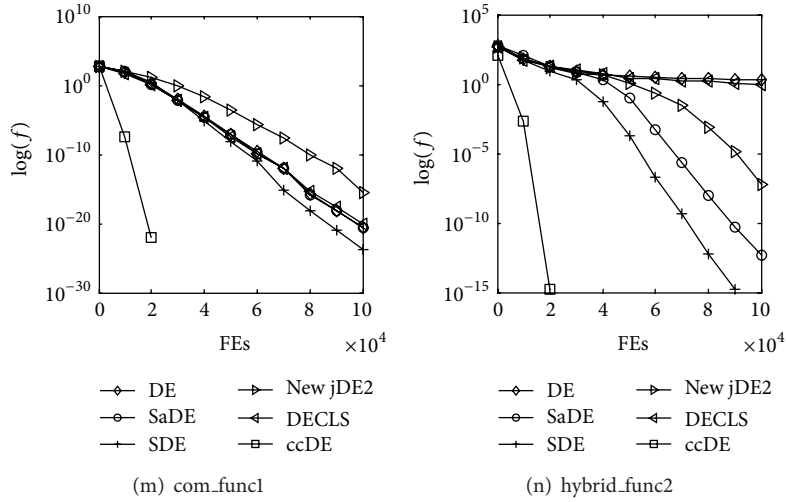
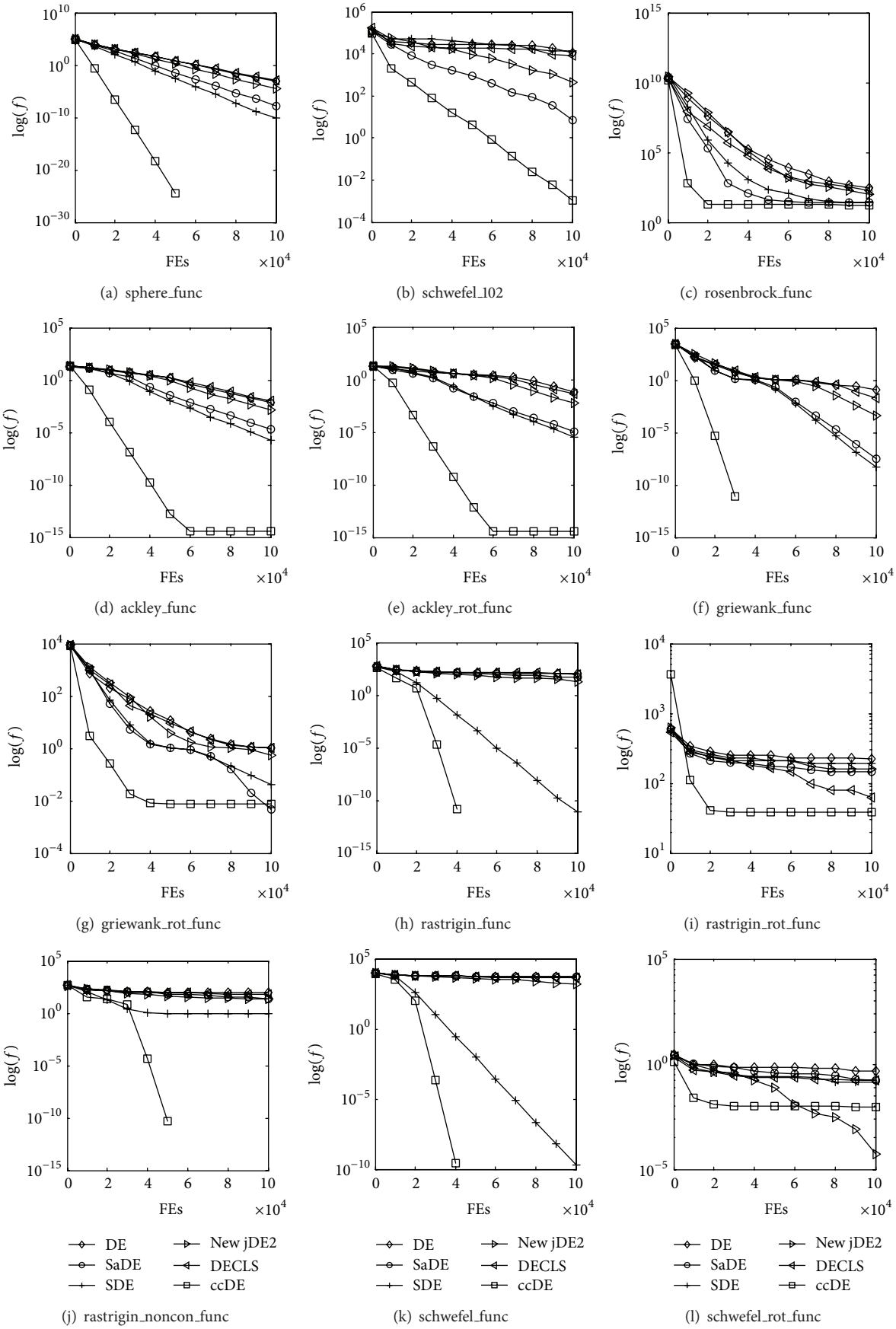


Figure 2: The median convergence characteristics of canonical DE, SaDE, SDE, jDE, DECLS, and ccDE on 10D problems.

Table 4: The mean and standard deviations of optimum (30D).

Algorithms		DE	SaDE	SDE	jDE	DECLS	ccDE
f_1	Mean	$6.81E-04$	$6.37E-09$	$9.15E-11$	$2.51E-05$	$2.27E-03$	**0.00E + 00**
	Std	$1.30E-04$	$2.04E-09$	$4.80E-11$	$6.56E-06$	$6.58E-04$	$0.00E+00$
f_2	Mean	$1.84E+04$	$3.57E+00$	$1.52E+04$	$1.81E+03$	$1.15E+04$	**3.55E − 03**
	Std	$3.51E+03$	$2.44E+00$	$3.34E+03$	$1.40E+03$	$1.43E+03$	$3.03E-03$
f_3	Mean	$1.62E+02$	$2.47E+01$	$6.13E+01$	$6.73E+01$	$2.88E+02$	**2.10E + 01**
	Std	$8.47E+01$	$3.37E-01$	$3.39E+01$	$3.09E+01$	$1.24E+02$	$3.27E+00$
f_4	Mean	$7.23E-03$	$1.14E-05$	$1.74E-06$	$1.46E-03$	$1.36E-02$	**3.55E − 15**
	Std	$9.88E-04$	$3.19E-06$	$4.36E-07$	$2.98E-04$	$2.95E-03$	$0.00E+00$
f_5	Mean	$3.69E-02$	$7.29E-06$	$4.30E-06$	$3.71E-03$	$6.02E-02$	**3.55E − 15**
	Std	$1.37E-02$	$1.65E-06$	$5.37E-07$	$4.80E-04$	$1.23E-02$	$0.00E+00$
f_6	Mean	$8.54E-02$	$8.93E-08$	$1.05E-09$	$2.35E-04$	$5.88E-02$	**0.00E + 00**
	Std	$8.65E-02$	$7.58E-08$	$1.91E-10$	$5.71E-05$	$4.76E-02$	$0.00E+00$
f_7	Mean	$1.03E+00$	$1.48E-02$	$1.55E-02$	$5.34E-01$	$1.06E+00$	**1.15E − 02**
	Std	$8.64E-03$	$1.37E-02$	$7.12E-03$	$5.27E-02$	$2.00E-02$	$5.12E-03$
f_8	Mean	$1.14E+02$	$4.38E+01$	$1.22E-11$	$1.99E+01$	$1.11E+02$	**0.00E + 00**
	Std	$7.59E+00$	$4.60E+00$	$6.93E-12$	$3.56E+00$	$3.93E+00$	$0.00E+00$
f_9	Mean	$2.43E+02$	$1.32E+02$	$1.79E+02$	$1.83E+02$	$8.50E+01$	**4.11E + 01**
	Std	$7.30E+00$	$7.33E+00$	$1.14E+01$	$1.74E+01$	$1.41E+01$	$2.15E+01$
f_{10}	Mean	$7.52E+01$	$2.81E+01$	$6.00E-01$	$1.98E+01$	$7.33E+01$	**0.00E + 00**
	Std	$8.89E+00$	$2.61E+00$	$5.48E-01$	$1.18E+00$	$6.40E+00$	$0.00E+00$
f_{11}	Mean	$4.63E+03$	$1.37E+03$	$1.08E-10$	$1.65E+03$	$4.47E+03$	**0.00E + 00**
	Std	$2.47E+02$	$1.21E+02$	$3.63E-11$	$2.36E+02$	$9.80E+01$	$0.00E+00$
f_{12}	Mean	$8.53E+03$	$7.07E+03$	$6.56E+03$	$5.40E+03$	$7.83E+03$	**4.16E + 03**
	Std	$2.04E+02$	$1.30E+02$	$9.66E+02$	$1.42E+03$	$7.33E+02$	$3.59E+02$
f_{13}	Mean	$7.77E-05$	$9.51E-10$	$5.14E-11$	$6.63E-06$	$2.64E-04$	**0.00E + 00**
	Std	$2.45E-05$	$5.01E-10$	$2.05E-11$	$1.96E-06$	$7.18E-05$	$0.00E+00$
f_{14}	Mean	$4.20E+01$	$1.12E+01$	**1.72E + 00**	$1.24E+01$	$2.61E+01$	$6.23E+00$
	Std	$4.43E+01$	$2.39E-01$	$8.31E-01$	$9.51E-01$	$1.38E+01$	$1.05E+00$



(a) sphere_func (b) schwefel_102 (c) rosenbrock_func

(d) ackley_func (e) ackley_rot_func (f) griewank_func

(g) griewank_rot_func (h) rastrigin_func (i) rastrigin_rot_func

(j) rastrigin_noncon_func (k) schwefel_func (l) schwefel_rot_func

Figure 3: Continued.

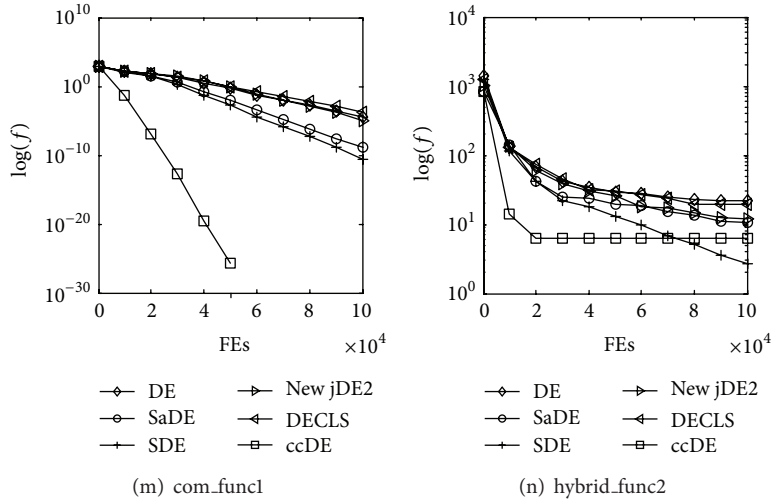


Figure 3: The median convergence characteristics of canonical DE, SaDE, SDE, jDE, DECLS, and ccDE on 30D problems.

Table 5: p values of the Wilcoxon matched-pairs signed-ranks test.

ccDE versus	$D = 10$	$D = 30$
DE	$1.0E-3$	$1.0E-3$
SaDE	$1.0E-3$	$1.0E-3$
SDE	$3.0E-3$	$8.0E-3$
jDE	$1.0E-3$	$1.0E-3$
DECLS	$1.0E-3$	$1.0E-3$

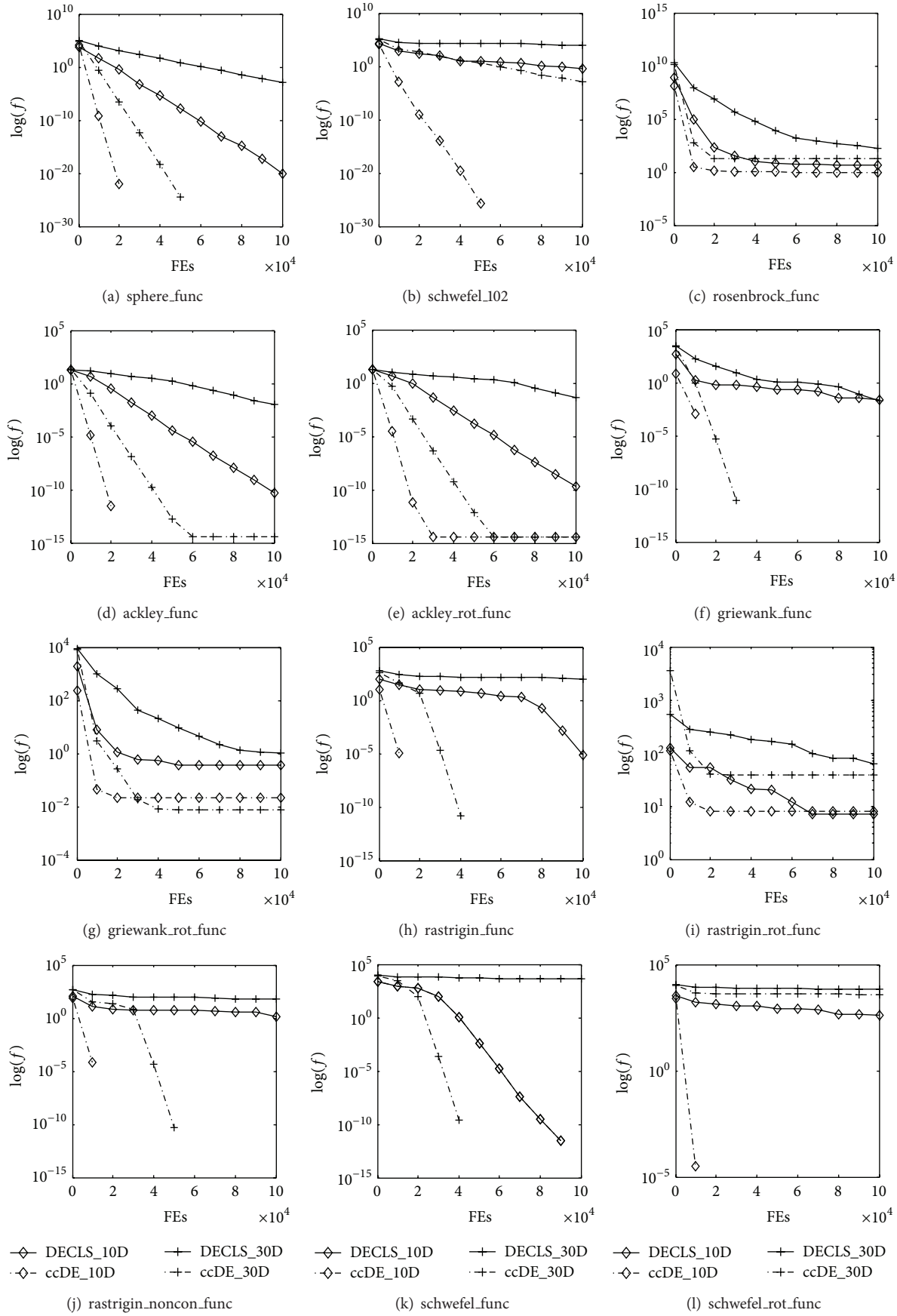
To illustrate the characteristics of the CLS operator, the cDE algorithm has been designed. The only difference between the cDE algorithm and ccDE algorithm is that cDE algorithm does not apply the CLS operator in Step 3 of the ccDE algorithm. Tables 6 and 7 illustrate the comparison of the mean, the standard deviation of the fitness value, and success rate between ccDE and cDE on 10D and 30D problems f_1–f_{14}, respectively. From Tables 6 and 7, it can be seen that the ccDE obtains a smaller mean value and a higher success rate than cDE for all problems. Accordingly, it is easier for the ccDE algorithm to avoid the immature convergence than the cDE algorithm, so it is easier to locate the global optima.

To illustrate the characteristics of orthogonal crossover operator, the bcDE algorithm has been designed. The only difference between the bcDE algorithm and ccDE algorithm is that bcDE algorithm applies the binomial crossover which is the same crossover strategy as canonic DE, while ccDE algorithm applies the orthogonal crossover. Figure 5 illustrates the convergence characteristics comparison in terms of the best fitness value between ccDE and bcDE on 10D and 30D problems f_1–f_{14}. From Figure 5, it can be seen that the convergence speed of ccDE is faster than bcDE for most of the 14 problems. Accordingly, the ccDE algorithm requires fewer FEs than the bcDE algorithm and therefore it has lower time complexity. In order to study the contribution of three mechanisms (chaotic local search, orthogonal crossover, and cellular automata) to the success of the algorithm, a unimodal function f_3 and multimodal function f_9 have been selected for the comparison of the global optimum distribution. The cDE, bcDE, and DECLS are short of chaotic local search, orthogonal crossover, and cellular automata mechanism, respectively.

Figure 6 shows the box plots of optimal solution distribution after 25 independent runs on f_3 and f_9 with 10D and 30D, respectively. From Figure 6, it can be seen that the optimal solution distribution of the canonical DE is the most scattered and ccDE algorithm is the most centralized, which indicates that three mechanisms contribute to the convergence in different extent. The abnormal values also appear when unimodal problem and multimodal problem are used to evaluate and compare those optimization algorithms including the cDE, bcDE, and DECLS algorithm. It can be found that those three mechanisms have the capability of getting away from local optimum to a certain extent. The biggest median of multimodal function is the cDE algorithm while the smallest is the ccDE algorithm, which indicates the chaotic local search mechanism is the most important to avoid local optimum.

5. Conclusions

This paper proposes the ccDE algorithm, which combines the DE algorithm with cellular automata to balance exploration and exploitation tradeoff of DE. In addition, the chaotic local search is employed as a tool to construct a search operator for optimization problems. At last, the binomial crossover operator in the canonical DE is replaced by the orthogonal crossover without crossover rate. Those mechanisms make the ccDE algorithm maintain population diversity while achieving faster convergence speed. Through comparing the performance of ccDE with the canonical DE and four DE variants on a set of 14 problems, the simulation results show that the ccDE algorithm has faster convergence speed and better capability of maintaining the population diversity, which can effectively avoid being trapped into



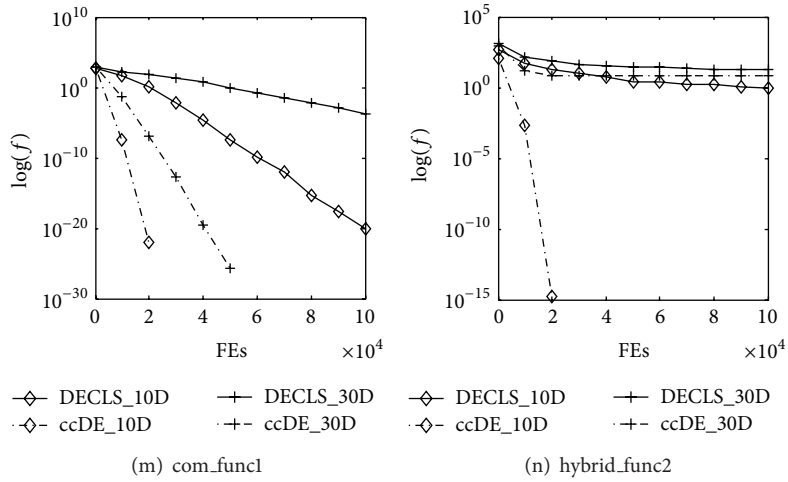
(a) sphere_func (b) schwefel_102 (c) rosenbrock_func

(d) ackley_func (e) ackley_rot_func (f) griewank_func

(g) griewank_rot_func (h) rastrigin_func (i) rastrigin_rot_func

(j) rastrigin_noncon_func (k) schwefel_func (l) schwefel_rot_func

Figure 4: Continued.



(m) com_func1 (n) hybrid_func2

Figure 4: The convergence characteristics between ccDE and DECLS on 10D and 30D problems.

Table 6: Results for 10D problems.

Problems	Algorithms					
	cDE			ccDE		
	Mean	Std	Success rate	Mean	Std	Success rate
f_1	**1.70E − 28**	**4.67E − 29**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_2	**3.78E − 11**	**7.90E − 12**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_3	2.12E + 01	1.09E + 01	63%	1.41E + 00	6.64E − 01	80%
f_4	1.46E − 13	1.36E − 14	87%	**0.00E + 00**	**0.00E + 00**	**100%**
f_5	**9.95E − 14**	**9.60E − 15**	**100%**	**3.55E − 15**	**0.00E + 00**	**100%**
f_6	**6.28E − 03**	**2.81E − 03**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_7	8.07E − 01	5.90E − 02	40%	4.92E − 02	2.59E − 02	63%
f_8	**0.00E + 00**	**0.00E + 00**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_9	1.40E + 02	4.07E + 01	50%	6.07E + 00	1.86E + 00	87%
f_{10}	**0.00E + 00**	**0.00E + 00**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_{11}	**0.00E + 00**	**0.00E + 00**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_{12}	3.53E + 03	2.37E + 02	27%	**0.00E + 00**	**0.00E + 00**	**100%**
f_{13}	2.82E − 03	1.26E − 03	83%	**0.00E + 00**	**0.00E + 00**	**100%**
f_{14}	2.08E + 01	3.05E + 00	67%	**0.00E + 00**	**0.00E + 00**	**100%**

Table 7: Results for 30D problems.

Problems	Algorithms					
	cDE			ccDE		
	Mean	Std	Success rate	Mean	Std	Success rate
f_1	**2.31E − 08**	**6.97E − 09**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_2	2.82E + 02	7.16E + 01	47%	3.55E − 03	3.03E − 03	98%
f_3	2.27E + 02	2.86E + 01	0%	2.10E + 01	3.27E + 00	53%
f_4	2.09E − 04	1.34E − 05	50%	**3.55E − 15**	**0.00E + 00**	**100%**
f_5	9.40E − 01	4.19E − 01	27%	**3.55E − 15**	**0.00E + 00**	**100%**
f_6	**5.95E − 06**	**8.16E − 07**	**100%**	**0.00E + 00**	**0.00E + 00**	**100%**
f_7	4.57E + 00	1.14E − 01	23%	1.15E − 02	5.12E − 03	63%
f_8	2.87E + 01	3.69E + 00	67%	**0.00E + 00**	**0.00E + 00**	**100%**
f_9	8.48E + 02	2.14E + 01	0%	4.11E + 01	2.15E + 01	50%
f_{10}	5.87E + 01	5.13E + 00	97%	**0.00E + 00**	**0.00E + 00**	**100%**
f_{11}	2.22E + 02	9.25E + 01	80%	**0.00E + 00**	**0.00E + 00**	**100%**
f_{12}	4.22E + 04	4.28E + 02	0%	4.16E + 03	3.59E + 02	0%
f_{13}	6.59E − 05	1.80E − 05	67%	**0.00E + 00**	**0.00E + 00**	**100%**
f_{14}	7.92E + 01	2.13E + 00	53%	6.23E + 00	1.05E + 00	80%

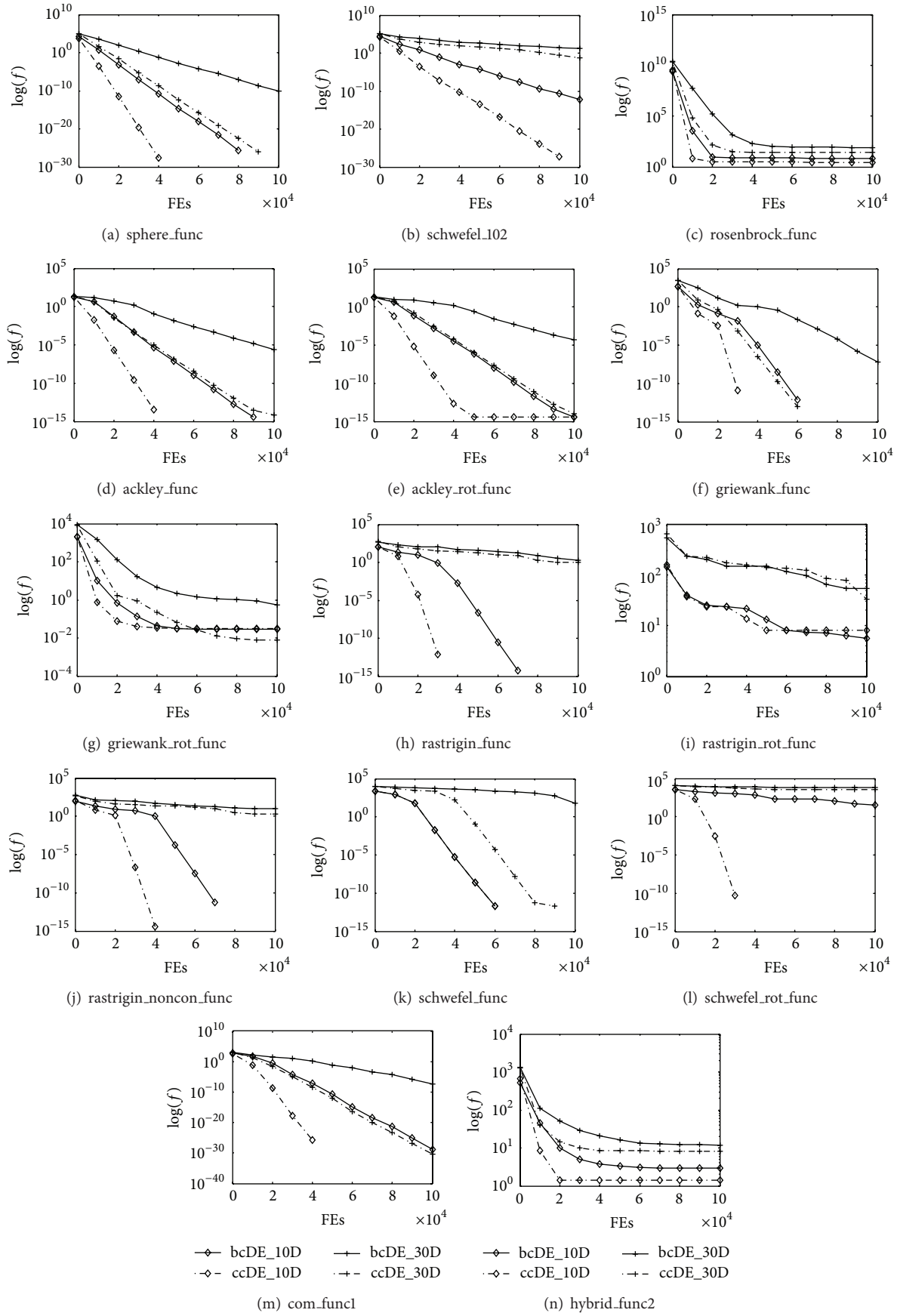


Figure 5: The convergence characteristics between bcDE and ccDE on 10D and 30D problems.

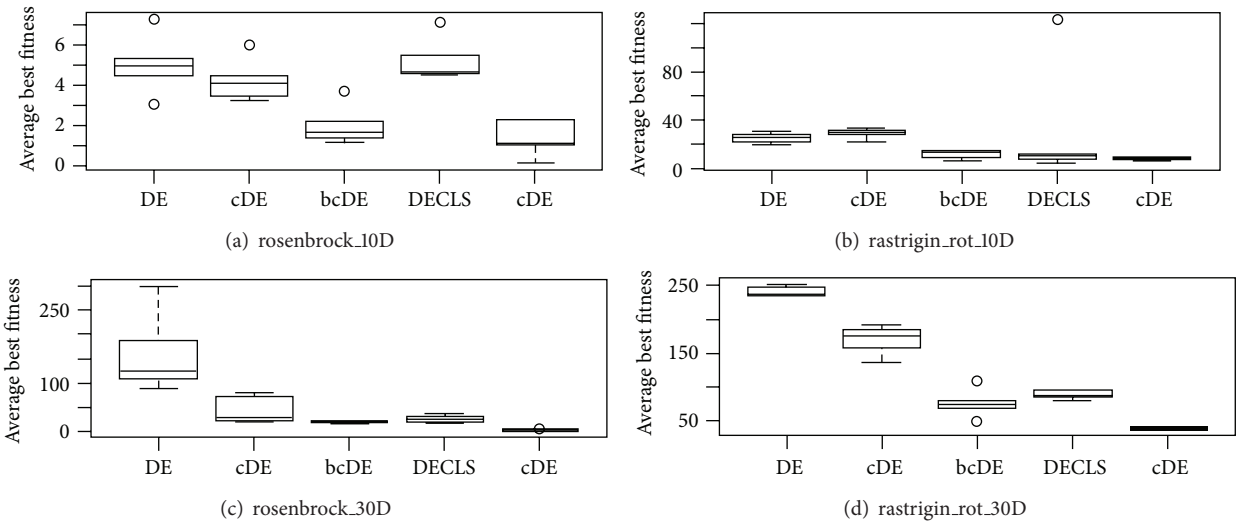


Figure 6: Box plots of optimal distribution for cDE, bcDE, DECLS, and ccDE.

the local optima. In the future, we will further study the ccDE algorithm for multiobjective optimization and dynamic optimization evolutionary mechanism.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (Grants nos. 61132003, 61171086, and 51267005), Jiangxi Province Science Foundation for Youths (20142BAB217008), and Shanghai Leading Academic Discipline Project (no. S30108). The authors thank Dr. P. N. Suganthan at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, for their help to improve the linguistic quality of this paper.

References

[1] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 209–225, 2014.

[3] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 126–142, 2005.

[4] M. Depolli, R. Trobec, and B. Filipič, "Asynchronous master-slave parallelization of differential evolution for multi-objective optimization," *Evolutionary Computation*, vol. 21, no. 2, pp. 261–291, 2013.

[5] E. den Heijer and A. E. Eiben, "Maintaining population diversity in evolutionary art using structured populations," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 529–536, June 2013.

[6] B. Dorronsoro and P. Bouvry, "Cellular genetic algorithms without additional parameters," *Journal of Supercomputing*, vol. 63, no. 3, pp. 816–835, 2013.

[7] C.-H. Cao, L.-M. Wang, and D.-Z. Zhao, "The research based on the discrete cellular ant algorithm in the geometric constraint solving," *Acta Electronica Sinica*, vol. 39, no. 5, pp. 1127–1130, 2011.

[8] Y.-M. Lu, M. Li, and L. Li, "The cellular genetic algorithm with evolutionary rule," *Acta Electronica Sinica*, vol. 38, no. 7, pp. 1603–1607, 2010.

[9] B. Lorenzo and S. Glisic, "Optimal routing and traffic scheduling for multihop cellular networks using genetic algorithm," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2274–2288, 2013.

[10] N. Noman and H. Iba, "Cellular differential evolution algorithm," in *AI 2010: Advances in Artificial Intelligence*, vol. 6464 of *Lecture Notes in Computer Science*, pp. 293–302, Springer, Berlin, Germany, 2010.

[11] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "CellularDE: a cellular based differential evolution for dynamic optimization problems," in *Adaptive and Natural Computing Algorithms*, vol. 6593 of *Lecture Notes in Computer Science*, pp. 340–349, Springer, Berlin, Germany, 2011.

[12] D. Jia, G. Zheng, and M. K. Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Information Sciences*, vol. 181, no. 15, pp. 3175–3187, 2011.

[13] J. V. Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, Ill, USA, 1996.

[14] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.

[15] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, pp. 153–177, 2012.

[16] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Computational Intelligence and Security*, Lecture Notes in Artificial Intelligence, pp. 192–199, Springer, 2005.

[17] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[18] 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.