doi:10.3772/j.issn.1006-6748.2017.04.012

Novel differential evolution algorithm with spatial evolution rules①

Ding Qingfeng (丁青锋)②, Qiu Xiang
(School of Electrical and Automation Engineering, East China Jiaotong University, Nanchang 330013, P. R. China)

Abstract

In order to reduce the pressure of parameter selection and avoid trapping into the local optimum, a novel differential evolution (DE) algorithm without crossover rate is proposed. Through embedding cellular automata into the DE algorithm, those interactions among vectors are restricted within cellular structure of neighbors while the cell own evolution, which may be used to balance the tradeoff between exploration and exploitation and then tune the selection pressure. And further more, the orthogonal crossover without crossover rate is used instead of the binomial crossover, which can maintain the population diversity and accelerate the convergence rate. Experimental studies are carried out on a suite of 7 bound-constrained numerical benchmark functions. The results show that the proposed algorithm has better capability of maintaining the population diversity and faster convergence than the classical differential evolution and several classic differential evolution variants.

Key words: differential evolution (DE), cellular automata, orthogonal crossover, balancing tradeoff, selective pressure

0 Introduction

Differential evolution (DE) algorithm[1] is a simple yet powerful evolutionary algorithm (EA) for global optimization in the continuous search domain[2,3]. However, the DE algorithm also has premature convergence, search stagnation and may be easily trapped into local optimum.

The spatial evolutionary algorithms have been proliferated over recent years, such as distributed evolutionary algorithms, cellular evolutionary algorithms (cEAs), etc. [4,5]. Thereinto cEAs are a sort of spatial structure-based EAs of discrete groups. A ratio of neighbor to population is the only parameters for contradiction evolutionary phenomena between exploration and exploitation to establish an adaptive dynamic cellular model and obtain the optimum behavior of efficiency and accuracy[6]. A hybrid optimization algorithm was proposed which combined the efforts of local search (vector learning) and cellular genetic algorithms for training recurrent neural networks (RNN's)[7]. A synchronous cellular genetic algorithm was proposed by bringing in synchronous mechanism to genetic algorithm[8]. The cellular DE algorithms with linear and compact neighbor structure were proposed but it did not consider the evolutionary process of cells[9,10]. The cellular DE algorithm was proposed to resolve dynamic optimization problems, which partitioned the search space into cells to find the local optimum yet not the cell own evolution[11]. In order to simulate the cellular evolutional condition more truly, a new cellular DE (cDE) algorithm of simulating the life phenomenon through embedding cellular automata (CA) into the DE algorithm is proposed.

The rest of this paper is organized as follows. In Section 1, the classical DE algorithm is introduced. Section 2 presents the proposed cDE with cellular evolutional process. Section 3 briefly illustrates the experimental results and discussions about the proposed cDE algorithm. Finally, the conclusions are demonstrated.

1 Classical DE

The classical DE algorithm is a population-based parallel iterative optimization algorithm which includes the initial stage and iterative evolutionary stage.

At the initial stage, the initial vector population is chosen randomly and should cover the entire parameter space. An evolutional population is $P^g(X) = [X_1^g, X_2^g, \cdots, X_i^g, \cdots, X_{NP}^g]$, where $X_i^g = [x_{i1}^g, \cdots, x_{ij}^g, \cdots, x_{iD}^g]$ denotes the ith vector of the gth generation, NP de-

① Supported by the National Natural Science Foundation of China (No. 61501186), the Jiangxi Province Science Foundation (No. 20171BAB202001) and the Visiting Scholar Foundation of Jiangxi Province Young and Middle - aged University Teachers' Development Program ([2016], No. 169).

② To whom correspondence should be addressed. E-mail: brandy724@sina.com

Received on Jan. 6, 2017

notes the population size and D denotes the dimension of the vector. And $g = [0, 1, \cdots, g_{\max}]$ and $g_{\max}$ denotes the maximum evolution generation. At the iterative evolutionary stage, each vector goes through a succession of iterative process including mutation, crossover and selection. The above iterative evolutionary stage will be repeated generation after generation until the termination criterion is satisfied.

2 Cellular differential evolution

2.1 The evolution mechanism of cEAs

CA proposed by von Neumann is a highly parallel computing model[12]. For vectors in cEAs are spread in a 2-D toroidal mesh and are only allowed to interact with their neighbors, the neighbor structure directly affects the algorithm performance, such as avoidance of local optimum and maintenance of the population diversity. The two classic cellular neighbor structures are linear (L) and compact (C) which are shown in Fig. 1.

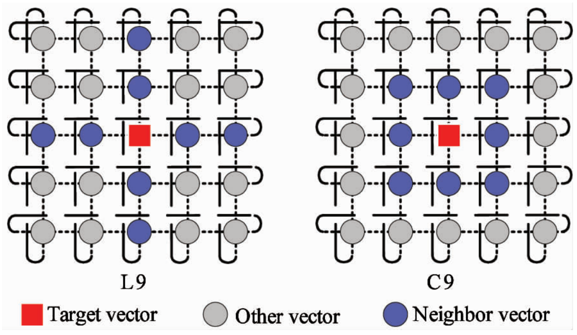

Fig. 1 Typical neighborhood structures of CA

The basic model of the evolution rule is as follows.

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

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

where S_t and S_{t+1} denote the cell status at step t and $t+1$, E_b is the number of an "alive" neighbor needing continuance in its status for an "alive" cell; F_k is the number of an "alive" neighbor needing resurrection for a "dead" cell.

The cellular survival space density is different for different evolution rules and affects the mutual restraint relationship between vectors. The three rules (**Rule**1: $E_b = \{2,3\}$, $F_k = \{3\}$, **Rule**2: $E_b = \{1,2,3,4\}$, $F_k = \{4,5,6,7\}$, and **Rule**3: $E_b = \{2,4,6,8\}$, $F_k = \{1,3,5,7\}$) are utilized for the simulation of stable, periodic and complex states which shall be caught in biological reproduction of confusion.

2.2 Orthogonal crossover operator

The aim of the cross operation is to pass the chunks with excellent properties in the vector to the vector of the next generation, making them excellent in parental vectors. However, the classical DE employs binomial or exponential crossover operator, which obviously can't obtain above objective in most cases. It is advisable to sample a small but representative sample of all combinations for test[13].

For the above purpose, $L_q(n^m)$ is employed as the orthogonal array for m factors and n levels, which has q rows in which every row represents a combination of levels[14]. In this paper, $n = 2$ represents that all factors have two levels: 0 and 1. Based on the level of every factor, the value of a trial vector will be selected from the target vector or its mutant vector in every dimension. $m = D$ is the dimension of the evolutional vector and q is the total number for completion of full test of D factors. The operation is described as the orthogonal crossover, which is shown in Fig. 2.

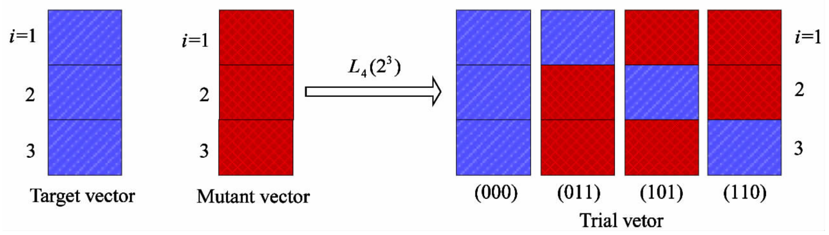

Fig. 2 Orthogonal crossover of a 3D vector

In Fig. 3 four selected combinations have been shown as black dots illustrating the orthogonality of a three-factor orthogonal array.

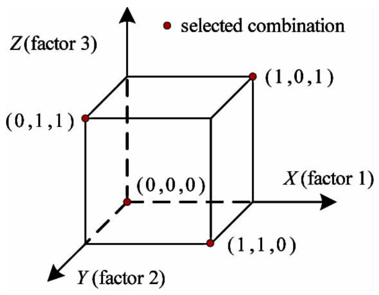

Fig. 3 Three-factor orthogonal array

2.3 Description of cDE algorithm

In the cDE algorithm, one cell value of the cellular automata represents the status of the evolutional vector with the same subscript: alive or dead, and only the "alive" vector can be evoluted. The size of the cellular automata is equal to the evolutional population. The proposed cDE algorithm is described as follows:

Algorithm: The Pseudocode of cDE

```
Step 1  Randomly initialize the vector value of the population: $P^g = \begin{bmatrix} X^g_{1,1} & \cdots & X^g_{1,j} & \cdots & X^g_{1,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X^g_{i,1} & \cdots & X^g_{i,j} & \cdots & X^g_{i,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X^g_{np,1} & \cdots & X^g_{np,i} & \cdots & X^g_{np,np} \end{bmatrix}_{np\times np}$,
    wherein $X^g_{i,j} = \{x^{g,1}_{i,j}, x^{g,2}_{i,j}, \cdots, x^{g,D}_{i,j}\}$ uniformly distributed in the range $[X_l, X_u]$,
    where $X_l = \{x^1_l, \cdots, x^D_l\}$ and $X_u = \{x^1_u, \cdots, x^D_u\}$ and $g = 0$.
Step 2  Randomly initialize the cellular value of CA: $C^g = \begin{bmatrix} c^g_{1,1} & \cdots & c^g_{1,j} & \cdots & c^g_{1,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c^g_{i,1} & \cdots & c^g_{i,j} & \cdots & c^g_{i,np} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c^g_{np,1} & \cdots & c^g_{np,j} & \cdots & c^g_{np,np} \end{bmatrix}_{np\times np}$,
    where $c^g_{i,j} = \begin{cases} 0, & rand < 0.5 \\ 1, & \text{otherwise} \end{cases}$.
Step 3  Select the orthogonal crossover table $L_Q(2^D) = [\alpha^T_1, \cdots, \alpha^T_q, \cdots, \alpha^T_Q]^T$ according to the dimension $D$ of the functions,
        wherein $\alpha_q = [\alpha_{q,1}, \cdots, \alpha_{q,d}, \cdots, \alpha_{q,D}]$.
Step 4  While (the value of $c^g_{i,j}$ corresponding to current target vector $X^g_{i,j}$ is 1, $i \in [1, np]$, $j \in [1, np]$)
    {
    Select three candidate vectors $\{X^g_{p1}, X^g_{p2}, X^g_{p3}\} = \mathbf{Select}(\mathbf{Neighbors}(X^g_{i,j}))$ according to specific CA neighborhood
        model.
    Execute the mutation operation to obtain the mutant vector: $V^g_{i,j} = \mathbf{Mutate}(X^g_{p1}, X^g_{p2}, X^g_{p3})$.
    Generate $Q$ chromosomes $[\hat{U}^T_1, \cdots, \hat{U}^T_q, \cdots, \hat{U}^T_Q]^T$ with $\hat{U}_q = [u_{q,1}, \cdots, u_{q,d}, \cdots, u_{q,D}]$,
        where $u_{q,d} = \begin{cases} x^{g,d}_{i,j}, & \text{if } \alpha_{q,d} = 1 \\ v^{g,d}_{i,j}, & \text{otherwise} \end{cases}$, then get $U^g_{i,j} = \underset{\{\hat{U}_1, \cdots, \hat{U}_q, \cdots, \hat{U}_Q\}}{\operatorname{argmin}} \; fitness\ (\hat{U}_q)$.
    Select the winner for the next generation vector: $X^{g+1}_{i,j} = \begin{cases} U^g_{i,j}, & \text{if } f(U^g_{i,j}) \leqslant f(X^g_{i,j}) \\ X^g_{i,j}, & \text{otherwise} \end{cases}$.
    }
Step 5  Update the cellular value of CA $C^{g+1}$ according to specific CA evolutional rule and update $g = g + 1$.
Step 6  If the termination criteria is satisfied, output the end result, otherwise jump to Step 4.
```

3 Experimental results and discussions

3.1 Description of benchmark functions

Experiments used to evaluate and compare several DE variants are conducted on a suite of 7 benchmark functions with different characteristics [2]. Functions f_1, f_2, f_4 and f_5 are shift for solving the problem that global optimum lies at the center of the search range. Functions f_3 and f_6 are rotated to avoid local optimum lying along the coordinate axes or suffer from global optimum lying at the center of the search range. Function f_7 is constructed by utilizing some basic problems to obtain one more challenging problem. For all functions, both 10-dimensional (10D) and 30-dimensional (30D) problems are tested. Table 1 outlines the function name, search range and global optimum of 7 benchmark functions.

Table 1 Descriptions of 7 benchmark functions

Function name	Search range	Global optimum
sphere _ func	[-100,100]	0
schwefel _ 102	[-100,100]	0
ackley _ rot	[-32,32]	0
griewank	[-600,600]	0
rastrigin	[-5,5]	0
schwefel _ rot	[-500,500]	0
com _ func1	[-5,5]	0

3.2 Parameter setting for comparison

Experiments in this paper are conducted 25 times independently on 7 benchmark functions to compare the proposed cDE with the classical DE, SDE, jDE, DE _ Zaharie and SaDE. The control parameters F, CR and NP of above six DE variants are set as used in Ref. [2].

For the proposed cDE algorithm, the orthogonal crossover employs the tailor-made array of the first 10-columns of the orthogonal list $L_{12}(2^{11})$ when the dimension of test problem is 10. In the orthogonal list $L_{12}(2^{11})$, 0 or 1 represents one dimension parameter of a trial vector which is selected from the target vector or its mutant vector of DE. The total number of experiments is 12. The orthogonal crossover is composed of three orthogonal lists at D = 10 side by side when the dimension of test problem is 30. The maximum number of function evaluations (FEs) is set to be 100 000 for all benchmark functions. For 12 combinations of levels are used in orthogonal crossover, the FEs of the cDE algorithm is set to be 100 000/12 for the sake of fairness. The cellular evolution rule in cDE in this paper uses Rule2[15]. The neighbor structure of vectors in cDE uses the optimum C9 neighbor model[9].

3.3 Comparison of the final solutions accuracy

In order to demonstrate the effects of CA on improving the performance of DE, cDE is compared with the classical DE algorithm and four kinds of DE variants, then the mean values and standard deviations (Std) of the best values are calculated by the results of 25 independent runs over 7 benchmark functions. Table 2 and Table 3 show the result of benchmark functions at D = 10 and D = 30 respectively, where the best value for each benchmark function has been shown in boldface.

From Table 2 and Table 3, it can be seen that the performance of cDE is superior to classical DE and four kinds of DE variants. The cDE algorithm obtains smaller mean values even theoretical optimum of all 7 benchmark functions at D = 10 and D = 30. Therefore it can be concluded that the mechanism of CA on improving the DE is significant.

Table 2 The means and standard deviations of 10D funcitons

Functions		DE	SaDE	SDE	jDE	DE _ Zaharie	cDE
f_1	Mean	2.80E-17	8.77E-16	4.02E-19	1.69E-13	1.02E-17	**0.00E+00**
	Std	9.27E-18	7.48E-16	2.82E-19	8.60E-14	3.64E-18	**0.00E+00**
f_2	Mean	1.96E+00	2.16E-13	2.29E-02	5.11E-04	1.04E+01	**7.10E-20**
	Std	5.39E-01	1.13E-13	1.38E-02	3.45E-04	8.55E-01	**3.34E-21**
f_3	Mean	1.14E-08	1.42E-08	3.68E-10	4.41E-07	1.40E-06	**3.55E-15**
	Std	4.40E-09	4.71E-09	3.66E-11	1.03E-07	8.44E-07	**0.00E+00**
f_4	Mean	1.01E-01	5.08E-02	1.04E-15	5.35E-03	2.01E-06	**0.00E+00**
	Std	3.52E-02	1.83E-02	6.16E-16	2.35E-03	3.66E-06	**0.00E+00**
f_5	Mean	1.34E-01	4.28E-01	**0.00E+00**	5.83E-05	**0.00E+00**	**0.00E+00**
	Std	1.45E-01	3.09E-01	**0.00E+00**	6.26E-05	**0.00E+00**	**0.00E+00**
f_6	Mean	7.40E+02	4.44E+02	9.45E-09	2.74E-08	8.40E+02	**2.10E-11**
	Std	3.18E+02	3.08E+02	1.31E-08	2.13E-08	1.95E+02	**5.26E-10**
f_7	Mean	2.17E-13	3.39E-16	2.75E-10	1.58E-12	6.61E-02	**0.00E+00**
	Std	4.84E-13	1.85E-16	6.15E-10	2.13E-12	9.65E-02	**0.00E+00**

Table 3 The means and standard deviations of 30D functions

Functions		DE	SaDE	SDE	jDE	DE _ Zaharie	cDE
f_1	Mean	6.81E-04	6.37E-09	9.15E-11	2.51E-05	2.23E-05	**1.12E-13**
	Std	1.30E-04	2.04E-09	4.80E-11	6.56E-06	3.31E-06	**9.24E-14**
f_2	Mean	1.84E+04	3.57E+00	1.52E+04	1.81E+03	1.88E+04	**5.90E-02**
	Std	3.51E+03	2.44E+00	3.34E+03	1.40E+03	4.89E+03	**7.52E-01**
f_3	Mean	3.69E-02	7.29E-06	4.30E-06	3.71E-03	5.80E-03	**1.07E-14**
	Std	1.37E-02	1.65E-06	5.37E-07	4.80E-04	9.03E-04	**3.36E-10**
f_4	Mean	8.54E-02	8.93E-08	1.05E-09	2.35E-04	6.44E-03	**0.00E+00**
	Std	8.65E-02	7.58E-08	1.91E-10	5.71E-05	3.12E-03	**0.00E+00**
f_5	Mean	1.14E+02	4.38E+01	1.22E-11	1.99E+01	1.05E+00	**0.00E+00**
	Std	7.59E+00	4.60E+00	4.45E-01	3.56E+00	1.58E+00	**0.00E+00**
f_6	Mean	8.53E+03	7.07E+03	6.56E+03	5.40E+03	7.59E+03	**3.45E+03**
	Std	2.04E+02	**1.30E+02**	9.66E+02	1.42E+03	2.30E+02	8.88E+02
f_7	Mean	7.77E-05	9.51E-10	5.14E-11	6.63E-06	2.89E-04	**8.77E-16**
	Std	2.45E-05	5.01E-10	2.05E-11	1.96E-06	2.16E-04	**7.34E-16**

Fig. 4 and Fig. 5 illustrate the convergence performance in terms of the median run of the best fitness value of the cDE, the classical DE, and four DE variants on 10D and 30D problems, respectively. It can be seen from Fig. 4 and Fig. 5 that cDE gets little different adaptation values from the classical DE and four DE variants at the initial stage. Furthermore, the convergence performances between cDE and some other DE

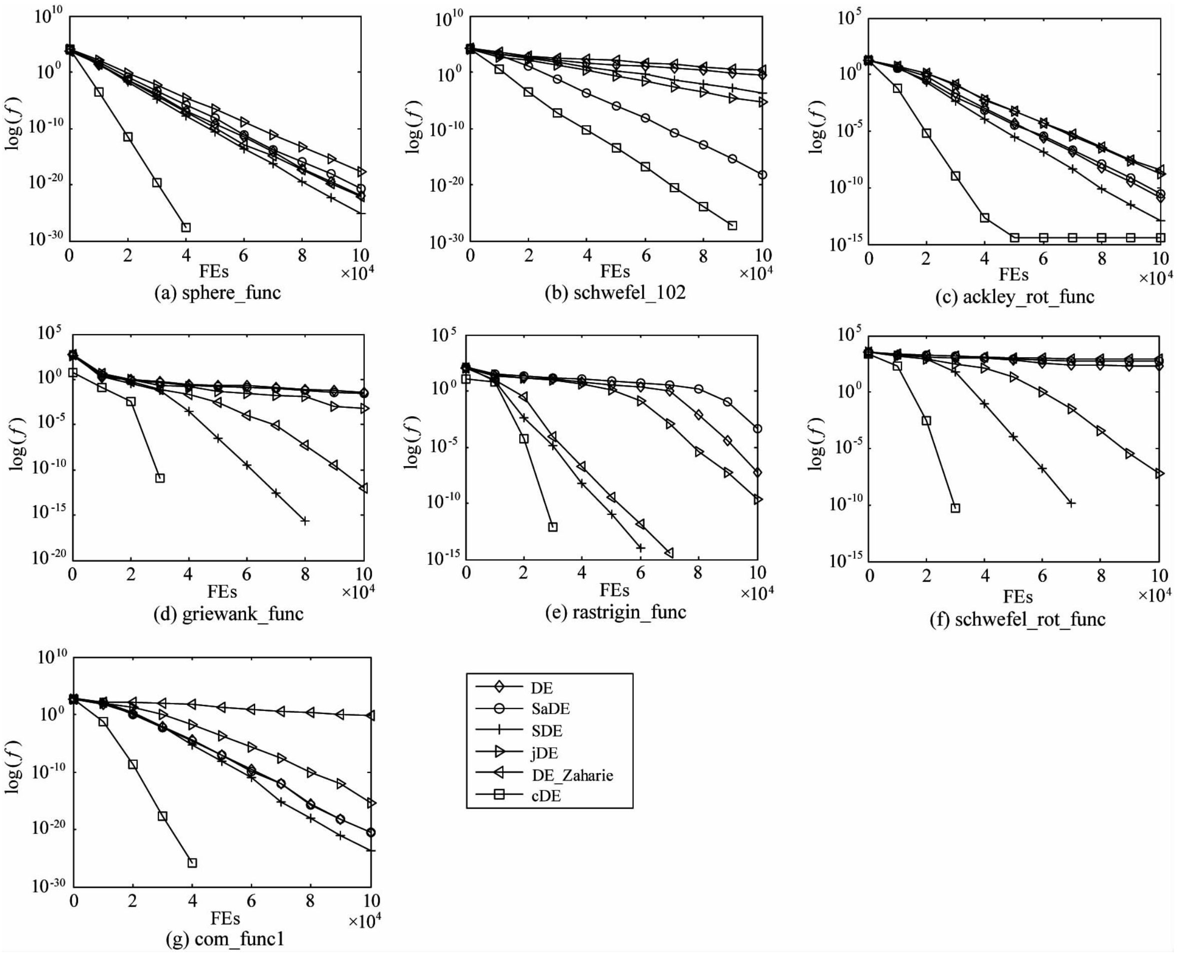

Fig. 4 The convergence performances of cDE in comparison with DE variants on 10D functions

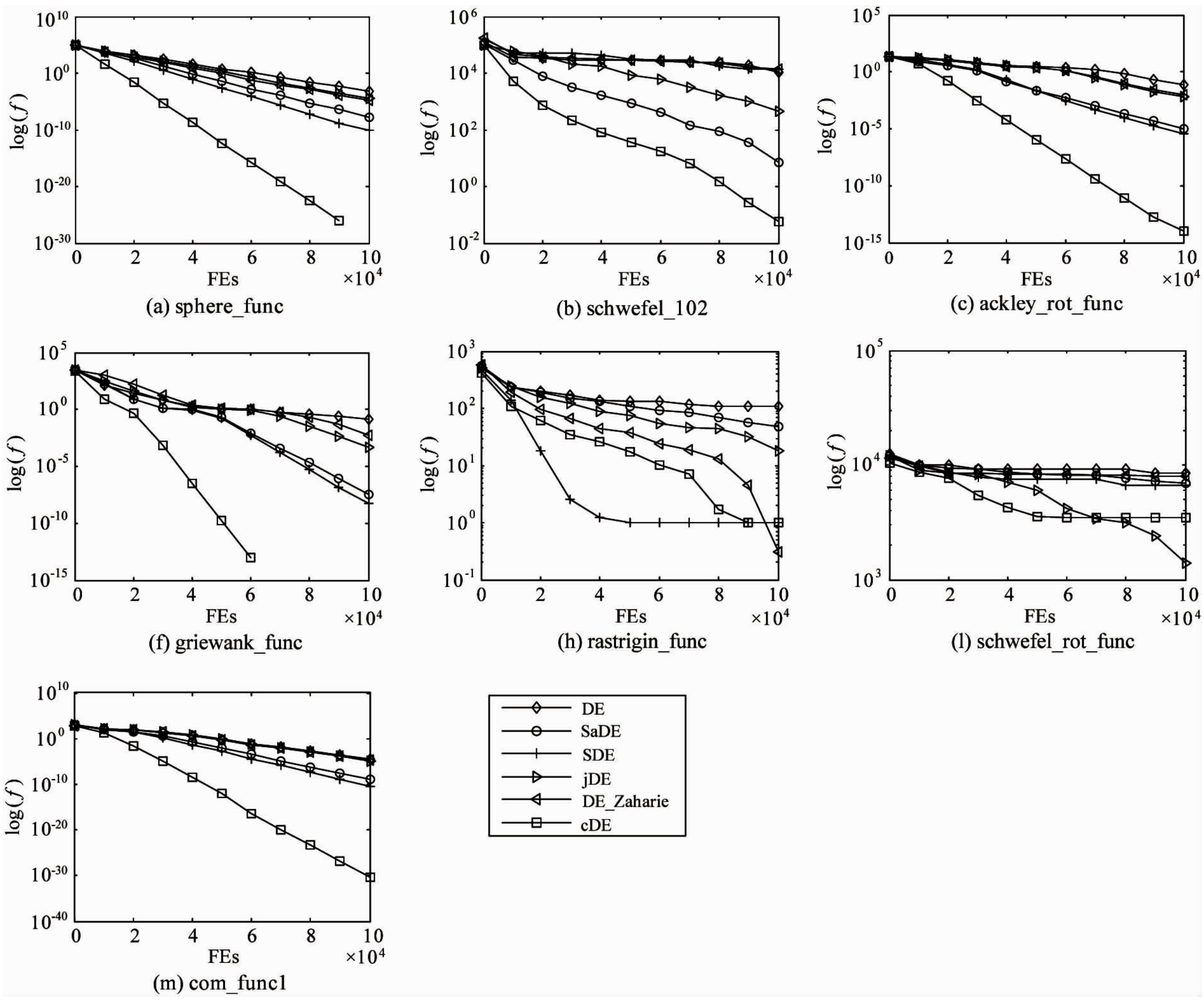

Fig. 5 The convergence performances of cDE in comparison with DE variants on 30D functions

variants in Fig. 4 show that the cDE always converges faster than others and has access to the optimum more easily than others on most of all 10D functions. Multiple benchmark functions for cDE algorithm have been converged end before 100,000 FEs. What's more, for the 30D functions, it is greatly difficult to find the global optimum for all DE variants. The cDE algorithm performs better on most problems in Fig. 5.

3.4 Wilcoxon matched-pairs signed-ranks test

In order to further illustrate the above conclusions, the Wilcoxon matched-pairs signed-ranks test is employed to compare cDE with the classical DE and four kinds of DE variants respectively. Table 4 is the Wilcoxon p-values of the mean data in Table 2 and Table 3.

As can be seen from Table 4, cDE outperforms the classical DE and four kinds of DE variants with the significance level $\alpha = 0.05$ considering independent matched-pairs comparisons at D = 10 and D = 30.

Table 4 p-values of Wilcoxon matched-pairs signed-ranks test

cDE vs.	D = 10	D = 30
DE	1.8E-2	1.8E-2
SaDE	1.8E-2	1.8E-2
SDE	2.7E-2	1.8E-2
jDE	1.8E-2	1.8E-2
DE _ Zaharie	2.8E-2	1.8E-2

3.5 Convergence analysis

To illustrate the effect of orthogonal crossover operator, the BicDE algorithm has been designed. The only difference between the BicDE algorithm and cDE algorithm is that BicDE algorithm uses the binomial crossover while cDE algorithm uses the orthogonal crossover.

Fig. 6 illustrates the convergence characteristics comparison in term of the best fitness value between cDE and BicDE on functions $f_1 \sim f_7$ at $D = 10$ and $D = 30$. From Fig. 6, it can be seen that the convergence rate of cDE is faster than BicDE for all 7 benchmark functions. Accordingly, the cDE algorithm requires smaller FEs than the BicDE algorithm and therefore it has smaller time complexity.

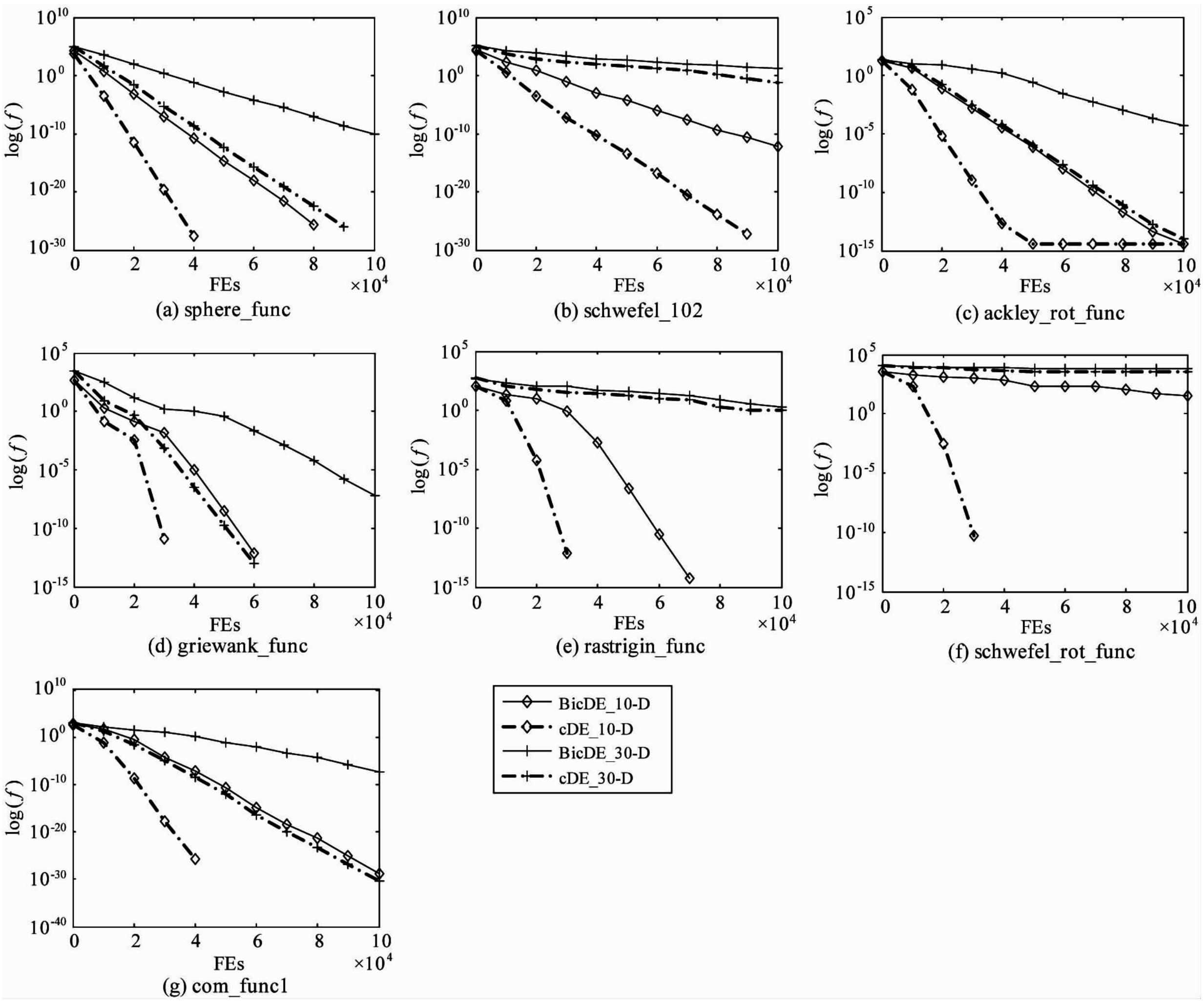

Fig. 6 The convergence characteristics of cDE vs BicDE for 7 benchmark functions

The above results confirm two conclusions. The first one is that cDE algorithm can tune the selection pressure by using local search within cellular neighbor structure instead of using its controlling parameters in the classical DE and plenty of cellular evolution rules ensure maintaining the diversity of evolutional population which can avoid falling into local optimum, so the probability of searching the global optimum will increase. The second one is that the orthogonal crossover operator provides repeated trials of multi-element for the cDE algorithm more easily to find the global optimum than those DE variants with binomial crossover and thus helps to accelerate the convergence speed.

4 Conclusions

A novel cDE algorithm without crossover rate is presented, in which those evolutional interactions among vectors are restricted within cellular neighbors while the cell is dynamic evolutionary. The parallel evolution mechanism and cellular neighbor stucture are applied to balance exploration and exploitation tradeoff of DE. The binomial crossover operator is replaced by the orthogonal crossover, which may maintain the population diversity and enhance convergence speed. Through comparing the convergence performance of cDE with the classical DE and four DE variants over a suit of 7 benchmark functions, the simulation results show that the cDE algorithm has better capability of maintaining the population diversity, which can avoid being trapped into the local optimum effectively and has faster convergence speed.

References

[1] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, 11(4): 341-359

[2] Das S, Suganthan P. Differential evolution: a survey of the state of the art. *IEEE Transactions on Evolutionary Computation*, 2011, 15(1): 4-31

[3] Jia L Y, He J X, Zhang C, et al. Differential evolution with controlled search direction. *Journal of Center South University*, 2012, 19(1): 3516-3523

[4] Alba E, Tomassini M. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(5): 443-462

[5] Michael O, Jimmy M. Predicting and monitoring the evolution of a coastal barrier dune system postbreaching. *Jouranl of Coastal Research*, 2013, 29(6a): 38-50

[6] Alba E, Dorronsoro B. The exploration/ exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 2005, 9(2): 126-142

[7] Kim W C K, Man W M, Wan C S. Adding learning to cellular genetic algorithms for training recurrent neural networks. *IEEE Transactions on Neural Networks*, 1999, 10(2): 239-252

[8] Zhang Y, Hu F, Liu Z, et al. Optimization design of truss structure based on synchronous cellular genetic algorithm. In: Proceedings of the 4th International Conference on Intelligent Control and Information Processing, Beijing, China, 2013. 94-97

[9] Noman N, Iba H. Cellular Differential Evolution Algorithm. Berlin Heidelberg: Springer, 2011. 293-302

[10] Dorronsoro B, Bouvry P. Differential Evolution Algorithms with Cellular Populations. Berlin Heidelberg: Springer, 2010. 320-330

[11] Noroozi V, Hashemi A, Meybodi M. CellularDE: A Cellular Based Differential Evolution for Dynamic Optimization Problems. Berlin Heidelberg: Springer, 2011. 340-349

[12] Von N, Burks A. Theory of Self-reproducing Automata. Urbana: University of Illinois Press, 1996. 1-6

[13] Zamani R. Integrating iterative crossover capability in orthogonal neighborhoods for scheduling resource-constrained projects. *Evolutionary Computation*, 2013, 21(2): 341-360

[14] Li H, Zhang L, Jiao Y C. Solution for integer linear bilevel programming problems using orthogonal genetic algorithm. *Journal of Systems Engineering and Electronics*, 2014, 25(3): 443-451

[15] Lu Y M, Li M, Li L. The cellular genetic algorithm with evolutionary rule. *Acta Electronica Sinica*, 2010, 38(7): 1603-1607

Ding Qingfeng, born in 1980. He received his Ph. D degree from the School of Communication and Information Engineering of Shanghai University in 2015 and his M. S. degree from the School of Electrical and Electronic Engineering of East China Jiaotong University in 2007. His research addresses the optimization technology in the receiver design of wireless communication, such as convex optimization and multi-objective optimization.