Differential Evolution Enhanced with Composite Population Information Based Mutation Operators

Jingliang Liao[1], Yiqiao Cai*[,1], Yonghong Chen[1], Tian Wang[1], Hui Tian[1]
[1]College of Computer Science and Technology
Huaqiao University
Xiamen, 361021, China
*Corresponding author
caiyq@hqu.edu.cn

JDIM
Journal of Digital Information Management

ABSTRACT: *Differential evolution (DE) is a simple and powerful evolutionary algorithm, which has been successfully used in various scientific and engineering fields. Generally, the base and difference vectors of the mutation operator in most of DE are randomly selected from the current population. Additionally, the population information is not fully exploited in the design of DE. In order to alleviate these drawbacks and enhance the performance of DE, this study presents a DE framework with Composite Population Information based mutation operator (DE-CPI) for global numerical optimization. In DE-CPI, the ring topology is employed to define a neighborhood for each individual and then the direction information with the neighbors is introduced into the mutation operator of DE. By this way, the composite population information, i.e., neighborhood and direction information, can be fully and simultaneously utilized in DE-CPI to guide the search of DE. In order to evaluate the effectiveness of the proposed method, DE-CPI is incorporated into the original DE algorithms, as well as several advanced DE variants. Experimental results clearly show that DE-CPI is able to enhance the performance of most of the DE algorithms studied.*

Subject Categories and Descriptors
I.5.3 [Clustering]; Algorithms **G.1.6 [Optimization]**

General Terms:
Evolutionary Algorithms, Neighborhood

Keywords: Differential Evolution, Neighborhood Information, Evolutionary Algorithm, Mutation Strategy

Received: 13 January 2015, Revised 20 February 2015, Accepted 27 February 2015

1. Introduction

Differential evolution (DE), proposed by Storn and Price [1], is a simple yet powerful evolutionary algorithm for global numerical optimization. It has many attractive characteristics, such as ease to use, simple structure, speediness and robustness. Due to these merits, DE has been extended to handle multi-objective, constrained, large-scale, dynamic, and uncertain optimization problems [2]. Furthermore, DE has been successfully used in diverse fields [2-4], such as chemical engineering, engineering design, pattern recognition, and so on.

In DE, there exist two main factors which significantly influence the performance of DE. The first one is the control parameters, i.e., population size *NP*, scaling factor *F*, and crossover rate *CR*, and the second one is the evolutionary operators, i.e., mutation, crossover and selection. In the literature about DE, there are many improved DE variants proposed during the last decade. Based on the reference [5], these advanced DE variants can be divided into two categories, DE with the additional components and DE with a modified structure. Modifications on DE in these variants mainly focus on introducing the self-adaptive strategies for the control parameters [6-9], devising the new mutation operators [10-13], developing the ensemble strategies [6][14-15], proposing the hybrid DE with other optimization algorithms [16] and population topology (multi or parallel population) [17], etc.

In the mutation operator of most DE variants, a mutant vector can be treated as the lead individual to explore the search space and generated by adding a difference vector to a base vector. We have observed, however, that these two vectors (i.e., base and difference vectors) in most of DE are usually selected randomly, which does not fully utilize the useful population information to guide the search.

In order to alleviate these drawbacks and enhance the performance of DE, we propose a new DE framework with *C*omposite *P*opulation *I*nformation based mutation operator (DE-CPI). In DE-CPI, a ring topology is firstly employed to obtain the neighborhood information by defining the neighbors for each individual. Then, the neighbors of each individual are partitioned into the better and worse groups according to their fitness values compared to that of it. After that, with respect to the base vector selected from the neighborhood of the current vector, the direction information is introduced into the mutation operator by selecting the vectors from the better and worse groups respectively to construct the difference vector. In this way, DE-CPI not only utilizes the information of neighboring individuals to exploit the regions of minima and accelerate convergence, but also incorporates the direction information of population to move the individuals to a promising area. Therefore, the composite population information, i.e., neighborhood and direction information, can be fully and simultaneously utilized in DE-CPI to guide the search of DE.

To evaluate the effectiveness of the proposed method, DE-CPI is applied to six original DE algorithms, as well as several advanced DE variants. Extensive experiments have been carried out on a set of benchmark functions. Through the extensive experimental study, the results show that DE-CPI is able to enhance the performance of most of the DE algorithms studied.

The main contributions of this study include the following:

- Both neighborhood and direction information, as the composite population information, are utilized fully and simultaneously to select the base and difference vectors for mutation, which is beneficial to guide the search of DE.
- DE-CPI provides a simple and effective way for enhancing the exploration ability of DE by combining the neighborhood and direction information of population.
- By keeping the simple structure of DE, DE-CPI is very simple and can be easily applied to other advanced DE variants.
- Extensive experiments have been conducted to show that DE-CPI can improve the performance of most of the DE algorithms studied in this paper.

The rest of this paper is organized as follows: In Section 2, the original DE algorithm is introduced. Section 3 briefly reviews some related work. The proposed DE-CPI is presented in detail in Section 4. In Section 5, the experimental results are reported. Finally, the conclusions are drawn in Section 6.

2. DE

DE is for solving the numerical optimization problem [1]. Without loss of generality, we consider the optimization problem to be minimized is $f(X)$, $X \in R^D$ and D is the dimension of the decision variables. DE evolves a population of NP vectors representing the candidate solutions. Each vector is denoted as $X_{i,G} = \{x^1_{i,G}, x^2_{i,G}, .., x^D_{i,G}\}$, where $i = 1, 2, ..., NP$, NP is the size of the population and G is the number of current generation.

2.1 Initialization

In DE, the initial population should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds. That is, the j^{th} parameter of the i^{th} individual is initialized by:

$$x^j_{i,G} = L_j + rand(0,1) \times (U_j - L_j) \quad (1)$$

where $rand(0,1)$ represents a uniformly distributed random number within the range [0, 1], and L_j and U_j represents the lower and upper bounds of the j^{th} variable respectively.

2.2 Mutation

After initialization, DE employs the mutation strategy to generate a mutant vector $V_{i,G}$ with respect to each individual $X_{i,G}$ (called target vector) in the current population. For example, several frequently used mutation strategies in the literature are listed as follows:

- DE/rand/1

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) \quad (2)$$

- DE/rand/2

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) + F \times (X_{r4,G} - X_{r5,G}) \quad (3)$$

- DE/best/1

$$V_{i,G} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) \quad (4)$$

- DE/best/2

$$V_{i,G} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}) \quad (5)$$

- DE/current-to-best/

$$V_{i,G} = X_{i,G} + F \times (X_{best,G} - X_{i,G}) + F \times (X_{r1,G} - X_{r2,G}) \quad (6)$$

- DE/rand-to-best/1

$$V_{i,G} = X_{r1,G} + F \times (X_{best,G} - X_{r1,G}) + F \times (X_{r2,G} - X_{r3,G}) \quad (7)$$

The indices r_1, r_2, r_3, r_4, r_5 are mutually exclusive integers that randomly generated within the range $[1, NP]$ and, which are also different from the index i. $X_{best,G}$ is the best

individual vector at generation G, and the mutation factor F is a positive control parameter for scaling the difference vector. Their more details can be found in [1-2].

2.3 Crossover

After the mutation phase, crossover operator is applied to each pair of $X_{i,G}$ and $V_{i,G}$ to generate a trial vector $U_{i,G}$. There are two kinds of crossover scheme: binomial and exponential. The binomial crossover is widely used, which can be defined as follows:

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j} & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand}; \\ x_{i,G}^{j} & \text{otherwise,} \end{cases} \quad (8)$$

where $CR \in [0, 1]$ is called the crossover rate. j_{rand} is a randomly chosen integer in the range [1, *D*]. If $u_{i,G}^{j}$ is out of the boundary, we randomly reinitialized it within the range $[L_j, U_j]$.

2.4 Selection

The selection operator selects the better one from each pair of $X_{i,G}$ and $U_{i,G}$ according to their fitness values for the next generation. The selection operator is given by

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(\mathrm{U}_{i,G}) \le f(\mathrm{X}_{i,G}); \\ X_{i,G} & \text{otherwise.} \end{cases} \quad (9)$$

3. Related Work

In this section, we focus on the related work on how the population information, especially neighborhood and direction information, has been utilized in the mutation operator of DE to improve its performance.

As the salient feature of DE, the mutation operator has been studied in various ways. In these DE variants, population information are often in focus and used in mutation to enhance the exploration ability of DE. In the literature, neighborhood and direction information are more widely and successfully used.

3.1 Neighborhood Information

The neighborhood concepts are usually used to improve the performance of DE. There are two main types of neighborhood information: one relies on the population topology and the other relies on the geographical locations on the fitness landscape. More details about the neighborhood concepts utilized in DE could be found in [18].

With the population topology, the neighbors of each individual do not necessary lie in the vicinity of its topological region in the search space. Different from the original DE algorithm, many DE variants utilize the neighborhood information with the structured population. In these DE variants, the individuals for the mutation strategies are selected according to a neighbor list constructed from the structured population. There are twoThere are two main canonical kinds of structured population in DE, i.e., distributed DE (dDE) [15 – 17] and cellular DE (cDE) [18 – 20]. In [12], a neighborhood-based mutation operator is employed by using the ring topology. In [25], the self-adaptive DE is modified by using a ring neighborhood topology. In [26], the bare bones DE is proposed by employing the concept of index neighborhoods in DE. Recently, five population topologies have been introduced in DE to improve its performance in [17].

With the geographical locations on the fitness landscape, the neighborhood information is derived from the dynamics of the population during the evolutionary process. In the DE variants with the geographical locations, the individuals for the mutation operator are selected from the vicinity of its topological region in the search space. In [18], a proximity-based DE framework (ProDE) is proposed by using an affinity matrix based on the Euclidean distance to select the individuals for mutation. For improving the performance of DE, the learning-enhanced DE (LeDE) is proposed in [27]. In LeDE, the neighborhoods of each individual involved in the intra-cluster learning strategy are defined based on the identified clusters.

3.2 Direction Information

In DE, the difference vector in the mutation operator is important for guiding the search and is often constructed in a random manner. In order to overcome this problem of DE, some works are proposed by exploiting the direction information for constructing the difference vector.

In [28], a trigonometric mutation DE (TDE) is proposed with a probabilistic triangle mutation strategy that incorporates the direction information into DE. In [29], a new mutation strategy, DE/rand/±mean, is proposed. In this strategy, the population is partitioned into two sub-populations according to the mean fitness value of all individuals. Then two vectors are randomly selected from the better sub-population and the worse one respectively to generate the different vector. In [30], a classification-based self-adaptive DE is proposed by using the direction information with the current best solution and the best previous solution of each individual. For the multi-objective optimization, there are several DE variants that also use the direction information in mutation to improve the performance [31-32]. Recently, a new DE framework, DE with neighborhood and direction information (NDi-DE), is proposed by explicitly introducing three types of direction information into mutation [11]. In NDi-DE, the three types of direction information play different roles to guide the search, and different mutation strategy equips with different type of direction information based on its search characteristic [11].

4. DE-CPI

As mentioned above, the neighborhood and direction information can be utilized to improve the performance of DE, but they are not fully and simultaneously exploited in the evolutionary process of DE. Therefore, we propose a new mutation operator based on the composite population

information (CPI), i.e., neighborhood and direction information, to enhance the exploration ability of DE. Furthermore, the complete framework, DE-CPI, is also algorithmically illustrated.

4.1 CPI Based Mutation Operator

In DE-CPI, the neighborhood and direction information, as the composite population information, are used simultaneously in the mutation operator. To implement CPI based mutation operator, we need to address two issues: First, how to define a neighborhood for each individual of population? Second, how to incorporate the direction information into mutation with the defined neighbors?

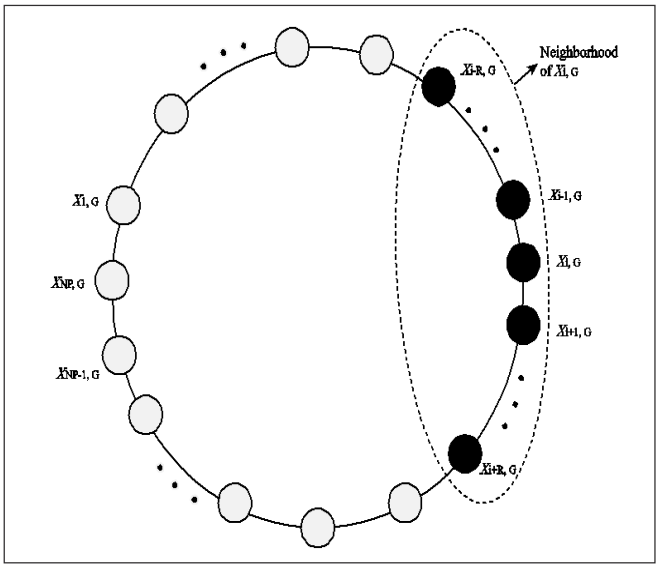

Figure 1. Ring topology of neighborhood in DE-CPI

4.1.1 Ring Topology-based Neighborhood

For the first issue, a generalized ring topology with *NP* vertexes is employed to define a neighborhood for each individual. Specifically, with the ring-topology, all the individuals of population are organized on a ring topology with respect to their indices. For each individual X_i, a neighborhood of radius R is defined based on the ring topology. That is, the neighbors of X_i is constructed by the individuals $X_{i-R},\ldots, X_i,\ldots, X_{i+R}$, where $R = p\times NP$, p is the neighborhood radius proportion. Therefore, each individual of population is with 2*R* neighbors. The ring topology of neighborhood in DE-CPI is illustrated in Figure 1. Although various neighborhood topologies, e.g., distributed, cellular, random topology, are proposed for DE [17], some initial experimental studies show that the ring topology can obtain the better and more robust results. The detailed comparisons of the DE-CPI variants with different neighborhood topologies will be studied in our future work.

4.1.2 Mutation with Direction Information

For the second issue, based on the defined neighbors with ring topology, the direction information is incorporated into mutation by selecting the vectors from the neighbors to construct the difference vector. Firstly, for each individual X_i, the base vector for mutation is randomly selected from the neighborhood of X_i[1]. Secondly, according to the fitness of the selected base vector, all the neighbors of X_i are partitioned into the better and worse groups. Finally, the terminal point of the difference vector (e.g., X_{r2} for DE/rand/1 in Eq. (2)) is selected from the better group and the start point (e.g., X_{r3} in Eq. (2)) is selected from the worse group. In this way, a difference vector that directing at the better solution from the worse one is constructed to guide the search.

4.2 The Framework of DE-CPI

By incorporating the CPI based mutation operator into DE, the complete framework of DE-CPI with the DE/rand/1 strategy (denoted as DE-CPI/rand/1) is shown in **Algorithm 1** where the differences with respect to DE/rand/1 are highlighted with "*". It is clear that the proposed DE-CPI framework only affects the mutation step of the original DE, hence it could be directly and easily applied to most of the DE mutation strategies.

In the application of DE-CPI for the mutation operators that incorporate the best individual (e.g., DE/best/1, DE/current-to-best/1, DE/rand-to-best/1, etc.), the definition of the best individual involved in mutation is redefined. That is, the best individual of population is replaced by the best vector in the neighborhood of the target individual in the mutation operator. In addition, when constructing the difference vector, if the base vector is the best vector in the neighborhood (e.g., DE/best/1), two neighbors of the base vector are randomly selected and the difference vector is constructed by directing the search from the worse neighbor to the better neighbor.

Compared with the original DE algorithm, the additional computation of DE-CPI depends on CPI based mutation operator. During one generation, the construction of difference vector for each target vector will take $R = p\times NP$ times of comparison with the base vector. Therefore, the complexity of CPI based mutation operator is $O(R)$. Since the complexity of the original DE algorithm is $O(G_{\max}\times NP\times D)$, where G_{max} is the maximal number of generation, the total complexity of DE-CPI is $O(G_{\max}\times NP\times \max(D,R))$.

5. Simulation Result

In order to evaluate the performance of DE-CPI, 25 benchmark functions from CEC 2005 [33] are used as the test suite.

[1] If the base vector is the best-so-far vector or the target vector, we do not need to select it in this way. The selection for the best-so-far base vector will be described in the following section.

Algorithm 1 DE-CPI/rand/1

```
1: Generate the initial population P^G, set G = 1, p = 0.1;
2: Evaluate the fitness for each individual in P^G;
3: While the terminated condition is not satisfied do
4:     For each individual X_{i, G} do
5:     *Randomly select the base vector X_{r1} from the neighborhood of X_{i, G};
6:        *Partition all the neighbors of X_{i, G} into a better and a worse
            groups by comparing with the fitness of X_{r1, G};
7:        *Randomly select X_{r2}, X_{r3} from the better and worse groups
                respectively;
8:         Use Eq. (2) to generate a mutant vector
9:         Use Eq. (8) to generate a trial vector;
10:       Use Eq. (9) to determine the survived vector;
11:    End For
12:    Set G = G + 1
13: End while
```

In this section, the benchmark functions are presented firstly. Secondly, the experimental setup is shown. Thirdly, the simulation results are analyzed and discussed.

5.1 Benchmark Functions

In this section, 25 benchmark functions are used, denoted as F1-F25, which are from the special session on real-parameter optimization of the 2005 IEEE Congress on evolutionary computation (CEC 2005) [33]. They can be categorized into four groups: unimodal functions (F1-F5), basic multimodal functions (F6-F12), expanded multimodal functions (F13-F14) and hybrid composition functions (F15-F25). More details of them can be found in [33].

5.2 Experimental Setup

In order to compare the performance between DE-CPI and its corresponding competitors, the same random initial population is used in this study. The parameters for all the experiments are set as follows unless a change is mentioned.

- Dimension of each function (*D*): 30 and 50.
- Population size (*NP*): 100.
- Mutation factor (*F*): 0.5.
- Crossover rate (*CR*): 0.9.
- Neighborhood radius proportion (*p*): 0.1.
- Number of runs (*NumR*): 25.
- Maximum number of function evaluations (*MNFs*): 10000×*D*.

In the experiments, the comparisons between six original DE algorithms (i.e., DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-best/1 and DE/rand-to-best/1) and their corresponding DE-CPI algorithms are conducted firstly. Then, we compare the performance of several advanced DE variants with the corresponding DE-CPI variants, including jDE [7], ODE [34], SaDE [6], CoDE [14], JADE [10] and MDE_pBX [35]. All the parameters of these DE variants are set as their original papers.

Furthermore, to show the significant differences among the competitors, several nonparametric statistical tests [36-37] are also carried out by the KEEL software [38]. The results of the single-problem Wilcoxon signed-rank test [36-37] at $\alpha = 0.05$ are firstly summarized in the last row of the tables as "*w/t/l*", which means that DE-CPI wins, ties and loses on *w*, *t* and *l* functions, compared with its corresponding competitor.

5.3 Comparison with Original DE Algorithms

In this section, six DE mutation operators (see Eq. (2) - (7)) are used in the experimental study. Due to the space limitation, the detailed experimental results of this simulation are not presented in this paper. The detailed results can be obtained from the corresponding author. The results of statistical tests for the functions at 30*D* and 50*D* are summarized in Table 1. The convergence graphs for some typical functions at 30*D* and 50*D* are also plotted in Figure 2 and Figure 3, respectively.

For the functions at 30*D*, Table 1 shows that DE-CPI can provide significantly better results than its corresponding original DE method in most of the test functions. Specifically, for DE/rand/1, DE-CPI is significantly better than the original DE on 12 out of 25 functions and is worse than it on 4 functions. For DE/rand/2, DE-CPI significantly wins on 22 functions and ties on 3 functions, compared with the original DE algorithm. DE-CPI/current-to-best/1

Algorithm at 30*D*	*w/t/l*	*R+*	*R-*	*p* -value	α = 0.05	α = 0.1
DE-CPI/rand/1 vs DE/rand/1	12/9/4	220.5	79.5	4.41E-02	+	+
DE-CPI/rand/2 vs DE/rand/2	22/3/0	320	5	5.96E-07	+	+
DE-CPI/current-to-best/1 vs DE/current-to-best/1	22/2/1	289	36	6.06E-04	+	+
DE-CPI/best/1 vs DE/best/1	22/2/1	296	4	8.34E-07	+	+
DE-CPI/best/2 vs DE/best/2	11/14/0	231.5	28.5	1.87E-02	+	+
DE-CPI/rand-to-best/1 vs DE/rand-to-best/1	6/10/9	140	160	1.00E+00	=	=
Algorithm at 50*D*	*w/t/l*	*R+*	*R-*	*p* -value	α = 0.05	α = 0.1
DE-CPI/rand/1 vs DE/rand/1	11/6/8	168.5	131.5	5.87E-01	=	=
DE-CPI/rand/2 vs DE/rand/2	22/3/0	323.5	1.5	1.49E-07	+	+
DE-CPI/current-to-best/1 vs DE/current-to-best/1	22/2/1	307	18	9.10E-05	+	+
DE-CPI/best/1 vs DE/best/1	22/2/1	298	2	3.58E-07	+	+
DE-CP/best/2 vs DE/best/2	13/11/1	264	36	5.68E-04	+	+
DE-CPI/rand-to-best/1 vs DE/rand-to-best/1	8/9/8	161.5	138.5	7.21E-01	=	=

+, *"* and = indicate DE-CPI is significantly better than, worse than and equal to its corresponding competitor overall based on the multi problem Wilcoxon signed-rank test, respectively.

Table 1. Results of the multi-problem Wilcoxon's test for DE-CPI versus the original DE algorithm for all the functions at 30*D*

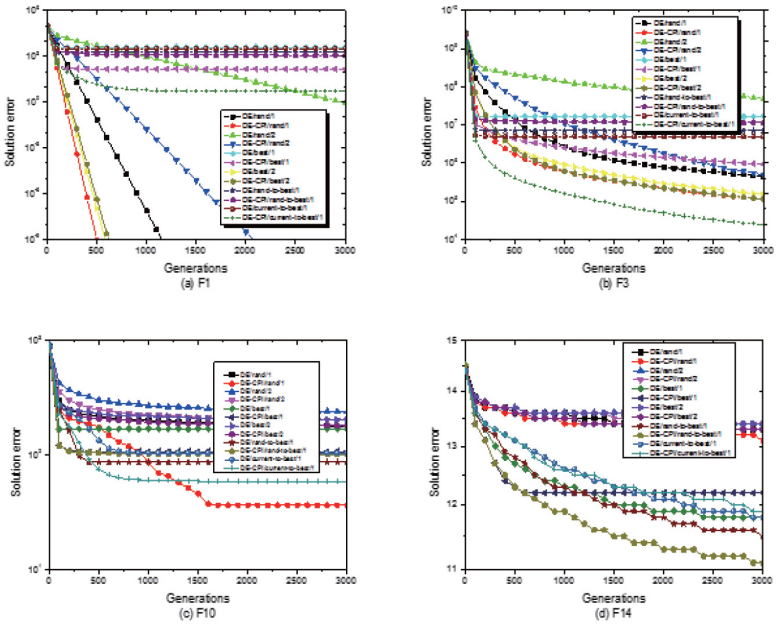

Figure 2. Convergence graphs of the original DE and the corresponding DE-CPI for the selected functions at $D = 30$

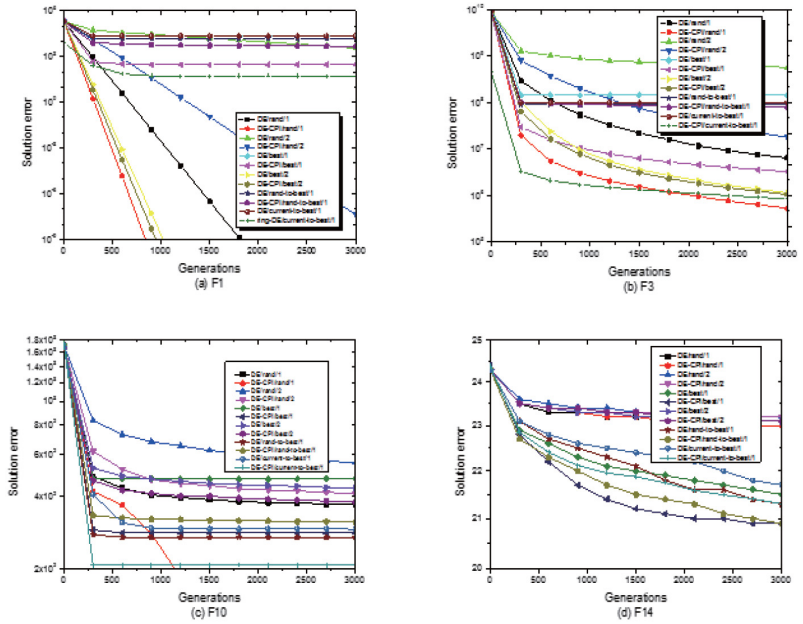

Figure 3. Convergence graphs of the original DE and the corresponding DE-CPI for the selected functions at $D = 50$

is significantly better than DE/current-to-best/1 on 22 functions, while DE-CPI/rand-to-best/1 is significantly better than DE/rand-to-best/1 on 6 functions. For DE/best/1 and DE/best/2, DE-CPI is significantly better than them on 22 and 11 functions, respectively.

For the functions at 50*D*, the results of Table 1 also show that DE-CPI is consistently superior to most of the corresponding DE algorithms. For DE/rand/1 and DE/rand/2, DE-CPI significantly outperforms them on 11 and 22 functions respectively. For DE/current-to-best/1 and DE/rand-to-best/1, DE-CPI is significantly better than them on 22 and 8 functions respectively. DE-CPI/best/1 is significantly better than DE/best/1 on 22 functions, while DE-CPI/best/2 is significantly better than DE/best/2 on 13 functions.

Algorithm at 30*D*	*w/t/l*	*R+*	*R-*	*p* -value	α = 0.05	α = 0.1
CoDE-CPI vs CoDE	21/3/1	288.5	11.5	5.30E-05	+	+
JADE-CPI vs JADE	3/20/2	164.5	160.5	9.46E-01	=	=
jDE-CPI vs jDE	5/19/1	248	77	2.03E-02	+	+
MDE_pBX-CPI vs MDE_pBX	11/11/3	219	106	1.24E-01	=	=
ODE-CPI vs ODE	9/14/2	208	117	2.16E-01	=	=
SaDE-CPI vs SaDE	10/14/1	209.5	90.5	8.54E-02	+	=
Algorithm at 50*D*	***w/t/l***	***R+***	***R-***	***p* -value**	**α = 0.05**	**α = 0.1**
CoDE-CPI vs CoDE	21/3/1	288	12	6.00E-05	+	+
JADE-CPI vs JADE	7/14/4	206	119	2.30E-01	=	=
jDE-CPI vs jDE	6/14/5	168	157	8.72E-01	=	=
MDE_pBX-CPI vs MDE_pBX	11/11/3	205	120	2.44E-01	=	=
ODE-CPI vs ODE	7/13/5	185	140	5.36E-01	=	=
SaDE-CPI vs SaDE	15/10/0	299.5	25.5	1.96E-04	+	+

Table 2. Results of the multi-problem Wilcoxon's test for DE-CPI versus the advanced DE variants for all the functions at 30*D*

+, ‘‘ and = indicate DE-CPI is significantly better than, worse than and equal to its corresponding competitor overall based on the multi problem Wilcoxon signed-rank test, respectively.

From Figure 2 and Figure 3, we can find that DE-CPI is better than the original DE algorithms in terms of the convergence speed for most of the selected functions.

Furthermore, to show the significant differences between DE-CPI and its corresponding DE algorithm, the multi-problem Wilcoxon signed-rank test is also carried out on all the problems at 30*D* and 50*D* [36-37]. The results are also shown in Table 1. For the functions at 30*D*, it is clear that DE-CPI can obtain the higher *R+* values than *R-* values in all the cases.

In addition, the *p* value in most of the cases are less than 0.05, which means that DE-CPI is significantly better than

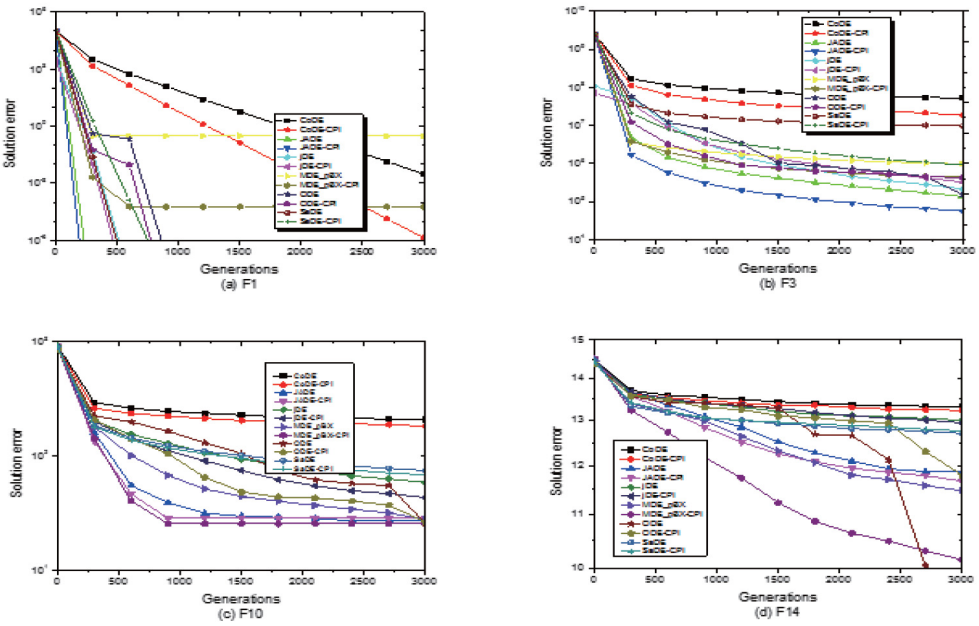

Figure 4. Convergence graphs of the advanced DE and the corresponding DE-CPI for the selected functions at *D* = 30

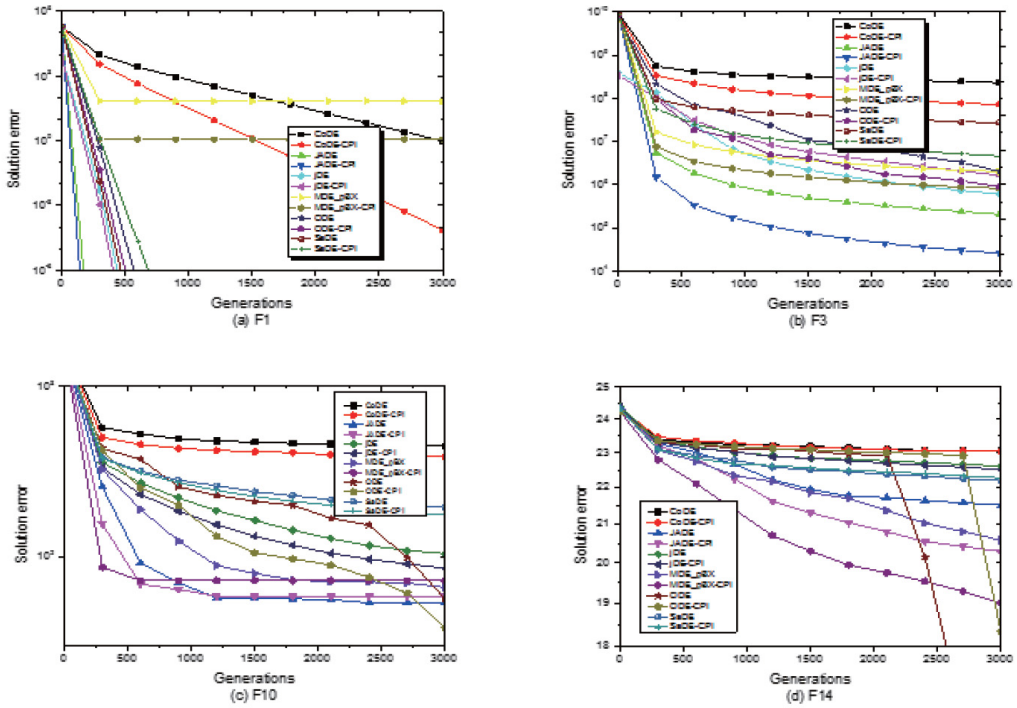

Figure 5. Convergence graphs of the advanced DE and the corresponding DE-CPI for the selected functions at $D = 50$

most of the original DE algorithms. For the functions at 50*D*, DE-CPI can consistently obtain the similar results as those in Table 1.

In general, these results indicate that DE-CPI is effective to improve the performance of most of the original DE algorithms studied.

5.4 Comparison with Advanced DE Variants

In order to evaluate the effectiveness of DE-CPI for the advanced DE variants, six recently proposed DE variants, jDE [7], ODE [34], SaDE [6], CoDE [14], JADE [10] and MDE_pBX [35], are studied in this section. These advanced DE variants integrate different kinds of modifications on DE, such as self-adaptive control parameters, new mutation operators, and ensemble strategies. They have different characteristics and can obtain very promising results. The results of statistical tests for the functions at 30*D* and 50*D* are summarized in Table 2. The convergence graphs for some typical functions at 30*D* and 50*D* are also plotted in Fig. 4 and Fig. 5 respectively. In addition, the detailed experimental results of this simulation can be obtained from the corresponding author.

From the results of functions at 30*D* in Table 2, we can find that DE-CPI exhibits substantial improvement for most of the DE variants. Specifically, for jDE, DE-CPI is significantly better than it on 5 functions and is worse than it on 1 function. For ODE, DE-CPI significantly outperforms it on 9 functions and is outperformed by it on 2 functions. For the DE variants with ensemble strategies, CoDE-CPI is significantly better than CoDE on 21 functions, while SaDE-CPI significantly outperforms SaDE on 10 functions. DE-CPI can obtain significantly improvements for MDE_pBX and JADE on 11 and 3 functions, respectively.

From the results of functions at 50*D* in Table 2, DE-CPI can also significantly enhance most of the DE variants on the test functions. For jDE and ODE, DE-CPI can obtain the significantly results on 6 and 7 functions, respectively. For SaDE and CoDE, DE-CPI is significantly better than them on 15 and 21 functions, respectively. For JADE and MDE_pBX, DE-CPI significantly outperforms them on 7 and 4 functions, respectively.

Fig. 4 and Fig. 5 also show that the convergence speed of DE-CPI is better than most of the corresponding advanced DE algorithms for most of the selected functions. Furthermore, the multi-problem Wilcoxon signed rank tests at $\alpha = 0.05$ and $\alpha = 0.1$ are conducted to show the significant differences between DE-CPI and its corresponding DE variant on all the problems at 30*D* and 50*D*, and the results are shown in Table 2 respectively. It is clear that DE-CPI can obtain the higher *R*+ values than *R*- values in all the cases. These results indicate that DE-CPI is better than its corresponding DE variants overall.

Summarily, the results of Table 2 and Figures. 4-5 show that DE-CPI can also bring the beneficial to most of the DE variants studied here.

5.5 Benefit of Composite Population Information

In DE-CPI, the composite population information, i.e., neighborhood and direction information, is exploited to guide the search of DE. In order to identify the benefit of the two kinds of population to DE-CPI, we consider two DE variants in this section. The first variant is DE-RING which only incorporates the neighborhood information of

ring topology into DE. The second variant is DE-DIR which only introduces the direction information into DE. In DE-RING, all the vectors for mutation are selected from the neighborhood of the target individual. In DE-DIR, the whole population is partitioned into the better and worse groups based on the fitness of the randomly selected base vector, and the difference vector is constructed by randomly choosing two vectors from the two groups respectively[2]. The experimental studies are carried out on the 25 functions at 30*D*, and four DE algorithms, i.e., DE/rand/1, DE/current-to-best/1, CoDE and SaDE, are used for comparison in this section. The results are shown in Table 3. The results of the single-problem Wilcoxon signed-rank test between the DE-CPI and the corresponding original DE algorithm at α = 0.05 are also summarized in Table 3.

From Table 3, it is clear that all the three variants, i.e., DE-CPI, DE-RING and DE-DIR, can obtain significantly better results than its corresponding original DE algorithm in most of the functions. Specifically, for DE/rand/2, DE-CPI, DE-RING and DE-DIR are significantly better than the original DE algorithm on 22, 12 and 21 functions respectively. For DE/current-to-best/1, DE-CPI significantly outperforms it on 22 functions, while DE-RING and DE-DIR are better than it on 20 and 14 functions respectively. For CoDE, DE-CPI, DE-RING and DE-DIR can obtain the significantly better results on 21, 14 and 13 functions respectively. For SaDE, DE-CPI, DE-RING and DE-DIR are significantly better than it on 10, 8 and 6 functions respectively.

In order to further show the effectiveness of CPI, the multi-problem Wilcoxon signed rank test between DE-CPI and DE-RING (DE-DIR) is also carried out and the results are shown in Table 4. It is interesting to find that DE-CPI obtains the higher *R*+ values than *R*- values in all the cases. Furthermore, the *p* values in most of the cases are less than 0.05, which means that DE-CPI is significantly better than DE-RING or DE-DIR in these cases. These results demonstrate that DE-CPI is better than its corresponding DE-RING and DE-DIR overall.

According to the results in Tables 3 and 4, we can obtain some interesting findings: 1) all of the neighborhood information, the direction information and CPI are beneficial to improving the performance of DE; 2) Compared with the single information (neighborhood or direction information), DE with CPI is more effectively for utilizing the population information to guide the search of DE.

5.6 Parameter Study

In DE-CPI, there is a control parameter (i.e., the neighborhood radius proportion *p*) that decides the neighborhood size of each individual. In order to investigate the influence of the *p* value on the performance of DE-CPI, the experiment studies on the 25 benchmark functions at 30*D* are conducted in this section. Here, DE/rand/1 is used for comparison and *p* is set to 0.05, 0.1, 0.2, 0.3 and 0.4. The results are shown Table 5. Furthermore, the results of the single-problem and the multi-problem Wilcoxon signed rank tests between DE/rand/1 and DE-CPI with different *p* value are also shown in Tables 5 and 6 respectively.

From Table 5, we can find that DE-CPI is better than DE/rand/1 in most of the cases overall. Specifically, in the case of $p = 0.1, 0.2, 0.3$ and 0.4, DE-CPI is significantly better than DE/rand/1 on 11, 11, 9 and 7 functions respectively and is worse than it on 5, 7, 5 and 4 functions respectively. When *p* = 0.05, DE-CPI significantly outperforms DE/rand/1 on 8 functions and is outperformed by it on 10 functions. According to the results of the statistical tests in Table 6, DE-CPI can obtain the higher *R*+ values than *R*- values in all the cases. It indicates that DE-CPI with different p values is better than DE/rand/1 overall. In addition, DE-CPI with $p = 0.1$ obtains the best results among all the cases, and DE-CPI with $p = 0.05$ and 0.4 obtain the worst and the second worst results respectively. The reasons may lie in: 1) When p is set to 0.05, the neighborhood size of each individual is small and the individuals belonging to the same neighborhood may quickly become similar to each other. This will make the population loses diversity and DE-CPI may have the problem of premature convergence; 2) When *p* is set to 0.4, the neighborhood size is large and the selection pressure for selecting parents from neighborhood will become too small. It will lead that the composite population information cannot effectively guide the search of DE with the neighbors.

In sum, the results of Tables 5 and 6 demonstrate that $p = 0.1$ is a good choice for DE-CPI when solving the benchmark problems studied here. Additionally, the value of $p \in [0.1, 0.3]$ has no significant influence on the performance of DE-CPI. In the future work, the adaptive or self-adaptive parameter control techniques, e.g., [6][8][10] and [39], will be studied for choosing the neighborhood size.

5.7 Application to Real-World Problems

In order to test the performance of DE-CPI on real-world problems, three problems from [40] and [41] are used in this section. The first two problems are selected from the CEC 2011 competition on testing EA on real-world numerical optimization problems [41]. They are parameter estimation for frequency modulated sound waves (denoted as FMP) and spread spectrum radar polly phase code design (denoted as SRP). FMP is a highly complex multi-modal problem with strong epistasis and SRP is with numerous local optima and has proven to be an *NP*-hard problem [41]. The last problem is systems of linear equations problem (denoted as LEP) which has proven to

[2] If the base vector is the best-so-far vector, the population does not need to be partitioned. Two vectors are randomly selected from the population, and the difference vector is constructed by directing the search from the worse vector to the better vector

Func.		DE-CPI/rand/2		DE-RING/rand/2		DE-DIR/rand/2		DE-CPI/current-to-best/1		DE-RING/current-to-best/1		DE-DIR/current-to-best/1
F1	+	**6.20e-016 5.28e-016**	+	1.55e-011 1.06e-011	+	2.03e-004 1.38e-004	+	**4.68e+000 1.20e+001**	+	1.69e+002 2.72e+002	+	3.62e+002 3.78e+002
F2	+	**2.10e-001 1.03e-001**	+	3.61e+000 1.69e+000	+	1.14e+003 7.69e+002	+	**1.71e-014 7.96e-014**	+	1.40e+003 5.55e+002	+	8.88e+002 8.38e+002
F3	+	**4.83e+005 1.73e+005**	+	1.01e+006 4.09e+005	+	1.55e+007 6.23e+006	+	**1.80e+004 9.53e+003**	+	2.51e+006 1.56e+006	=	6.02e+005 1.40e+006
F4	+	**1.88e+001 9.62e+000**	+	9.98e+001 3.87e+001	+	4.75e+003 2.01e+003	+	**1.56e-009 2.60e-009**	+	9.25e+001 8.75e+001	+	2.85e+000 9.79e+000
F5	+	**5.77e+002 1.30e+002**	+	1.28e+003 2.24e+002	+	4.78e+003 7.93e+002	+	**2.08e+002 1.91e+002**	+	3.34e+003 8.52e+002	+	2.66e+003 9.36e+002
F6	+	**7.76e+000 2.70e+000**	+	1.66e+001 1.06e+000	+	3.73e+001 2.23e+001	+	**3.61e+004 1.77e+005**	+	1.51e+007 1.88e+007	+	1.05e+007 1.61e+007
F7	+	**9.86e-004 2.76e-003**	+	4.76e+003 1.27e+001	+	5.31e+003 5.47e+001	+	**4.70e+003 0.00e+000**	+	4.75e+003 5.37e+001	+	4.70e+003 1.71e+000
F8	=	2.10e+001 3.85e-002	=	2.09e+001 5.80e-002	=	**2.09e+001 3.59e-002**	+	2.06e+001 1.93e-001	=	2.09e+001 5.63e-002	=	2.09e+001 4.47e-002
F9	+	**1.91e+002 9.88e+000**	=	1.94e+002 1.08e+001	+	2.07e+002 9.72e+000	+	**4.94e+001 1.59e+001**	+	6.09e+001 1.59e+001	+	5.27e+001 1.03e+001
F10	+	2.03e+002 1.04e+001	=	**2.02e+002 1.18e+001**	+	2.32e+002 1.20e+001	+	7.53e+001 2.36e+001	+	9.15e+001 2.42e+001	=	**6.23e+001 2.11e+001**
F11	=	**3.92e+001 1.34e+000**	=	3.95e+001 1.58e+000	=	3.96e+001 9.30e-001	-	1.73e+001 2.88e+000	+	2.31e+001 2.78e+000	-	**1.21e+001 2.14e+000**
F12	+	**1.45e+004 3.41e+004**	+	2.31e+005 1.46e+005	+	4.86e+005 6.45e+004	+	9.07e+003 8.64e+003	+	1.23e+004 7.58e+003	=	**1.88e+003 4.01e+003**
F13	+	**1.62e+001 1.47e+000**	=	1.69e+001 9.03e-001	+	1.95e+001 8.72e-001	+	3.39e+000 9.42e-001	=	**3.29e+000 1.07e+000**	=	5.91e+000 4.00e+000
F14	=	1.34e+001 1.72e-001	=	1.34e+001 1.66e-001	=	1.34e+001 1.62e-001	=	**1.16e+001 4.66e-001**	=	1.18e+001 3.26e-001	=	1.18e+001 4.36e-001
F15	+	3.96e+002 2.00e+001	=	4.00e+002 0.00e+000	-	**3.94e+002 2.97e+001**	=	**4.04e+002 3.69e+001**	=	4.29e+002 4.95e+001	=	4.24e+002 1.00e+002
F16	+	**2.27e+002 1.33e+001**	=	2.29e+002 1.02e+001	+	2.58e+002 1.18e+001	+	**8.11e+001 1.52e+001**	+	1.13e+002 2.91e+001	+	2.27e+002 1.85e+002
F17	+	**2.50e+002 1.03e+001**	=	2.55e+002 1.10e+001	+	2.92e+002 2.00e+001	+	**1.19e+002 8.97e+001**	+	1.37e+002 9.05e+001	+	2.43e+002 1.56e+002
F18	+	**9.03e+002 2.15e+001**	+	9.04e+002 2.16e+001	+	9.27e+002 2.46e+000	+	**8.94e+002 5.99e+001**	+	9.47e+002 6.19e+001	=	9.28e+002 1.19e+001
F19	+	9.08e+002 1.26e+000	=	**8.95e+002 3.58e+001**	+	9.26e+002 2.60e+000	+	**8.98e+002 6.29e+001**	+	9.23e+002 6.82e+001	=	9.22e+002 2.65e+001
F20	+	**8.99e+002 2.97e+001**	+	9.08e+002 1.10e+000	+	9.26e+002 3.01e+000	+	**8.95e+002 6.08e+001**	=	9.20e+002 7.63e+001	=	9.23e+002 2.26e+001
F21	+	5.00e+002 1.53e-005	=	**5.00e+002 6.23e-006**	+	5.00e+002 4.54e-005	+	**5.67e+002 1.91e+002**	+	7.82e+002 3.27e+002	+	8.16e+002 2.54e+002
F22	+	**9.22e+002 7.13e+000**	+	9.35e+002 7.12e+000	+	9.74e+002 1.39e+001	+	**9.27e+002 2.12e+001**	+	9.59e+002 2.04e+001	+	9.61e+002 2.35e+001
F23	+	**5.34e+002 1.41e-004**	=	5.34e+002 2.81e-004	+	5.34e+002 8.88e-004	+	**6.12e+002 1.86e+002**	+	9.07e+002 2.23e+002	+	8.37e+002 2.19e+002
F24	+	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	+	2.00e+002 1.01e-004	+	**2.00e+002 3.11e-006**	+	2.01e+002 3.68e+000	+	3.96e+002 2.14e+002
F25	+	**6.17e+002 1.98e+000**	+	1.69e+003 3.55e+000	+	1.70e+003 3.71e+000	+	**1.66e+003 6.93e+000**	+	1.68e+003 1.13e+001	+	1.66e+003 7.73e+000
w/t/l		22/3/0		12/13/0		21/3/1		22/2/1		20/5/0		14/10/1
Func.		**CoDE-CPI**		**CoDE-RING**		**CoDE-DIR**		**SaDE-CPI**		**SaDE-RING**		**SaDE-DIR**
F1	+	2.03e-003 4.71e-004	-	**1.77e-003 4.28e-004**	+	1.73e-003 3.28e-004	=	0.00e+000 0.00e+000	=	0.00e+000 0.00e+000	=	0.00e+000 0.00e+000
F2	+	**7.76e+003 1.19e+003**	+	9.70e+003 1.63e+003	+	8.46e+003 1.77e+003	+	**4.50e-003 2.03e-003**	+	1.58e+000 8.71e-001	+	6.23e-002 4.17e-002
F3	+	**4.07e+007 7.93e+006**	+	4.62e+007 7.33e+006	+	4.29e+007 6.37e+006	+	**9.21e+005 2.49e+005**	+	8.00e+006 1.95e+006	+	6.82e+006 1.83e+006
F4	+	**1.24e+004 2.02e+003**	+	1.55e+004 2.79e+003	+	1.35e+004 1.79e+003	+	5.52e+000 2.80e+000	+	3.58e+002 2.00e+002	-	**2.59e+000 1.43e+000**
F5	+	**4.55e+003 4.62e+002**	+	5.17e+003 3.86e+002	+	4.96e+003 4.85e+002	+	**8.86e+002 2.56e+002**	+	2.50e+003 3.34e+002	=	1.16e+003 6.42e+002
F6	+	3.53e+002 5.08e+001	+	3.73e+002 6.47e+001	-	**3.48e+002 6.59e+001**	+	**2.53e+001 2.01e+001**	+	5.23e+001 3.14e+001	=	2.94e+001 2.69e+001
F7	=	4.70e+003 0.00e+000	=	4.70e+003 0.00e+000	=	4.70e+003 0.00e+000	=	4.70e+003 0.00e+000	=	4.70e+003 0.00e+000	=	4.70e+003 7.05e-005
F8	=	2.09e+001 5.16e-002	=	**2.10e+001 3.57e-002**	=	2.09e+001 4.96e-002	=	2.09e+001 5.70e-002	=	2.09e+001 4.84e-002	=	2.09e+001 5.72e-002
F9	+	1.62e+001 1.40e+000	=	1.68e+001 1.72e+000	=	**1.60e+001 1.55e+000**	=	0.00e+000 0.00e+000	=	0.00e+000 0.00e+000	=	0.00e+000 0.00e+000
F10	+	**2.02e+002 1.13e+001**	+	2.07e+002 1.35e+001	+	2.03e+002 1.20e+001	+	6.82e+001 8.29e+000	+	7.53e+001 8.63e+000	-	**6.16e+001 7.96e+000**
F11	=	3.49e+001 1.34e+000	=	3.49e+001 9.55e-001	=	**3.47e+001 1.59e+000**	=	2.78e+001 1.54e+000	=	**2.71e+001 1.73e+000**	=	2.78e+001 1.12e+000
F12	+	**1.54e+005 2.19e+004**	+	1.46e+005 1.95e+004	+	1.55e+005 1.88e+004	+	**1.50e+004 5.24e+003**	=	1.64e+004 4.16e+003	+	1.83e+004 4.71e+003
F13	+	9.76e+000 7.76e-001	-	**9.70e+000 6.74e-001**	+	9.87e+000 7.53e-001	=	2.43e+000 2.22e-001	=	2.37e+000 1.95e-001	=	**2.31e+000 1.84e-001**
F14	-	1.33e+001 1.83e-001	=	1.33e+001 1.34e-001	=	1.33e+001 2.21e-001	=	1.28e+001 1.79e-001	=	1.27e+001 2.38e-001	=	1.28e+001 1.73e-001
F15	+	3.96e+002 4.45e+001	+	4.00e+002 9.50e-003	-	**3.76e+002 6.55e+001**	=	**2.45e+002 1.40e+002**	=	2.61e+002 1.08e+002	+	3.23e+002 9.27e+001
F16	+	**2.26e+002 1.12e+001**	+	2.29e+002 1.43e+001	+	2.31e+002 1.52e+001	=	9.62e+001 1.84e+001	=	1.01e+002 1.32e+001	=	**9.05e+001 2.06e+001**
F17	+	**2.55e+002 1.35e+001**	+	2.59e+002 1.21e+001	+	2.55e+002 1.46e+001	=	1.57e+002 1.90e+001	=	1.55e+002 2.19e+001	-	**1.45e+002 2.94e+001**
F18	+	9.07e+002 2.87e-001	+	9.08e+002 3.42e-001	-	**9.07e+002 2.67e-001**	-	8.78e+002 4.99e+001	-	8.23e+002 4.73e+001	+	8.98e+002 3.70e+001
F19	+	9.07e+002 3.22e-001	+	9.08e+002 3.78e-001	-	**9.07e+002 2.91e-001**	=	8.87e+002 4.46e+001	-	**8.37e+002 5.45e+001**	+	9.02e+002 3.08e+001
F20	+	**9.07e+002 2.33e-001**	+	9.08e+002 3.81e-001	+	9.07e+002 3.08e-001	=	8.97e+002 3.65e+001	-	**8.46e+002 5.70e+001**	=	8.98e+002 3.68e+001
F21	+	5.00e+002 1.11e-004	-	**5.00e+002 8.19e-005**	-	5.00e+002 8.33e-005	=	5.00e+002 8.81e-006	=	5.00e+002 0.00e+000	=	5.00e+002 0.00e+000
F22	+	**9.22e+002 9.40e+000**	+	9.50e+002 7.81e+000	+	9.31e+002 8.31e+000	+	9.27e+002 7.24e+000	+	9.37e+002 7.66e+000	-	**9.11e+002 9.44e+000**
F23	+	5.34e+002 4.01e-004	-	5.34e+002 3.05e-004	-	**5.34e+002 2.29e-004**	+	5.34e+002 9.05e-005	=	5.34e+002 1.58e-004	-	**5.34e+002 7.71e-005**
F24	+	2.00e+002 2.43e-004	-	2.00e+002 1.85e-004	-	**2.00e+002 1.53e-004**	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000
F25	+	1.66e+003 2.30e+000	-	**1.66e+003 2.26e+000**	+	1.66e+003 2.32e+000	+	1.66e+003 3.68e+000	+	1.67e+003 3.21e+000	=	**1.66e+003 3.26e+000**
w/t/l		21/3/1		14/5/6		13/5/7		10/14/1		8/14/3		6/14/5

Table 3. Mean and standard deviation of the best error values obtained by DE-RING, DE-DIR and DE-CPI on all the functions at 30*D*

Algorithm	*w/t/l*	*R+*	*R-*	*p* -value	α = 0.05	α = 0.1
DE-CPI/rand/2 vs DE-RING/rand/2	12/13/0	286	39	7.82E-04	+	+
DE-CPI/rand/2 vs DE-DIR/rand/2	21/3/1	206	19	1.07E-04	+	+
DE-CPI/current-to-best/1 vs DE-RING/current-to-best/1	20/5/0	297	3	2.50E-05	+	+
DE-CPI/current-to-best/1 vs DE-DIR/current-to-best/1	14/10/1	294	6	3.70E-05	+	+
CoDE-CPI vs CoDE-RING	14/5/6	314.5	10.5	3.20E-05	+	+
CoDE-CPI vs CoDE-DIR	13/5/7	273.5	26.5	3.09E-04	+	+
SaDE-CPI vs SaDE-RING	8/14/3	197.5	102.5	1.69E-01	=	=
SaDE-CPI vs SaDE-DIR	6/14/5	200.5	124.5	3.00E-01	=	=

Table 4. Results of the multi-problem Wilcoxon's test for DE-CPI versus DE-RING and DE-DIR for all the functions at 30*D*

Func.	DE/rand/1	DE-CPI/rand/1											
		p =0.05		p =0.1		p =0.2		p =0.3		p =0.4			
F1	**0.00e+000 0.00e+000**	-	3.20e-028 2.80e-028	-	1.28e-028 1.35e-028	-	9.49e-029 1.30e-028	-	1.35e-028 1.37e-028	-	1.17e-028 1.71e-028		
F2	7.13e-005 5.73e-005	-	6.06e-004 9.38e-004	+	**6.87e-013 1.81e-012**	+	**1.40e-012 2.23e-012**	+	**2.38e-010 4.77e-010**	+	**4.95e-010 6.08e-010**		
F3	4.45e+005 2.60e+005	=	3.73e+005 1.73e+005	+	**1.13e+005 5.92e+004**	+	**1.28e+005 7.36e+004**	+	**1.59e+005 9.89e+004**	+	**2.22e+005 1.28e+005**		
F4	2.12e-002 1.70e-002	-	4.71e+000 7.86e+000	+	**8.38e-006 2.65e-005**	+	**1.15e-005 4.60e-005**	+	**1.87e-005 3.25e-005**	+	**4.51e-005 7.94e-005**		
F5	6.65e+001 7.38e+001	-	2.25e+003 3.97e+002	-	4.75e+002 3.31e+002	+	**1.86e+001 5.14e+001**	+	**4.67e+000 7.94e+000**	+	**4.66e+000 6.33e+000**		
F6	2.80e+000 1.63e+000	-	1.15e+005 5.73e+005	-	2.39e+001 1.53e+001	+	**4.85e-001 1.32e+000**	+	**1.59e-001 7.97e-001**	+	**3.19e-001 1.10e+000**		
F7	3.94e-004 1.97e-003	-	4.70e+003 0.00e+000	-	2.12e-002 1.65e-002	-	4.70e+003 0.00e+000	+	**4.70e+003 7.05e-005**	-	4.70e+003 3.59e-004		
F8	2.10e+001 4.30e-002	+	**2.09e+001 6.09e-002**	=	2.09e+001 5.47e-002	=	2.09e+001 5.36e-002	=	2.10e+001 4.07e-002	=	2.09e+001 4.72e-002		
F9	1.30e+002 2.76e+001	+	**2.26e+001 1.13e+001**	+	**2.86e+001 9.62e+000**	+	**2.27e+001 6.28e+000**	+	**1.98e+001 1.12e+001**	+	**5.79e+001 4.48e+001**		
F10	1.79e+002 1.18e+001	+	**5.96e+001 1.05e+001**	+	**3.62e+001 1.01e+001**	+	**1.36e+002 6.19e+001**	=	1.77e+002 1.34e+001	=	1.78e+002 9.91e+000		
F11	3.96e+001 1.42e+000	+	**2.79e+001 3.93e+000**	+	**2.74e+001 6.53e+000**	=	3.90e+001 2.93e+000	=	3.94e+001 1.09e+000	=	3.96e+001 1.20e+000		
F12	1.32e+003 1.79e+003	-	5.23e+003 3.26e+003	=	2.65e+003 3.38e+003	=	9.49e+002 1.10e+003	=	8.69e+002 1.74e+003	=	8.92e+002 1.42e+003		
F13	1.51e+001 9.53e-001	+	**3.25e+000 8.33e-001**	+	**3.23e+000 8.09e-001**	+	**3.60e+000 2.24e+000**	+	**1.02e+001 4.22e+000**	+	**1.32e+001 2.61e+000**		
F14	1.33e+001 1.44e-001	+	**1.19e+001 6.36e-001**	+	**1.31e+001 1.54e-001**	=	1.33e+001 1.30e-001	=	1.33e+001 1.94e-001	=	1.33e+001 1.62e-001		
F15	4.04e+002 2.00e+001	=	3.85e+002 5.59e+001	=	3.93e+002 6.33e+001	+	**3.64e+002 7.00e+001**	=	3.92e+002 8.12e+001	=	3.80e+002 5.78e+001		
F16	2.05e+002 9.51e+000	+	**1.00e+002 3.93e+001**	+	**5.99e+001 1.64e+001**	+	**1.14e+002 6.96e+001**	+	**1.90e+002 4.48e+001**	=	2.05e+002 1.94e+001		
F17	2.28e+002 1.95e+001	+	**1.10e+002 7.33e+001**	+	**5.60e+001 9.96e+000**	+	**2.07e+002 4.24e+001**	=	2.27e+002 5.58e+001	=	2.24e+002 1.95e+001		
F18	8.97e+002 2.91e+001	=	8.64e+002 6.26e+001	=	8.80e+002 5.08e+001	-	9.02e+002 2.14e+001	=	9.05e+002 1.66e+000	=	9.01e+002 2.11e+001		
F19	8.92e+002 3.46e+001	=	8.78e+002 5.98e+001	=	8.80e+002 5.08e+001	-	9.02e+002 2.13e+001	-	9.01e+002 2.10e+001	-	9.05e+002 1.39e+000		
F20	9.00e+002 2.09e+001	=	8.69e+002 6.28e+001	=	8.85e+002 4.86e+001	-	9.07e+002 2.04e+000	-	9.05e+002 1.53e+000	=	8.92e+002 3.48e+001		
F21	5.00e+002 0.00e+000	=	5.00e+002 1.25e-005	=	5.00e+002 0.00e+000	=	5.00e+002 0.00e+000	=	5.00e+002 0.00e+000	=	5.12e+002 6.00e+001		
F22	9.08e+002 9.46e+000	-	9.18e+002 1.46e+001	=	9.06e+002 1.14e+001	=	9.02e+002 1.21e+001	=	9.07e+002 1.00e+001	=	9.02e+002 1.13e+001		
F23	5.34e+002 9.99e-005	-	5.50e+002 8.13e+001	-	5.34e+002 1.13e-002	-	5.34e+002 1.06e-002	-	5.34e+002 3.48e-004	=	5.34e+002 3.28e-004		
F24	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000	=	2.00e+002 0.00e+000		
F25	6.13e+002 1.54e+000	-	1.66e+003 4.64e+000	+	**2.16e+002 8.20e+001**	-	1.64e+003 1.17e+001	-	1.65e+003 8.95e+000	-	1.66e+003 9.00e+000		
w/t/l	-		8/7/10		11/9/5		11/7/7		9/11/5		7/14/4		

Table 5. Mean and standard deviation of the best error values obtained by DE/rand/1 and DE-CPI/rand/1 with different neighborhood size radius (p) on all the functions at 30D.

Algorithm	w/t/l	R+	R-	p-value	α = 0.05	α = 0.1
DE-CPI with p =0.05 vs DE/rand/1	8/7/10	169.5	155.5	8.40E-01	=	=
DE-CPI with p =0.1 vs DE/rand/1	12/9/4	220.5	79.5	4.25E-02	+	+
DE-CPI with p =0.2 vs DE/rand/1	11/7/7	217	108	1.39E-01	=	=
DE-CPI with p =0.3 vs DE/rand/1	9/11/5	187	113	2.82E-01	=	=
DE-CPI with p =0.4 vs DE/rand/1	7/14/4	185.5	114.5	3.02E-01	=	=

+, ‘‘ and = indicate DE-CPI is significantly better than, worse than and equal to the corresponding original DE overall based on the multi problem Wilcoxon signed-rank test, respectively.

Table 6. Results of the multi-problem Wilcoxon’s test for DE/rand/1 versus DE-CPI with different neighborhood size radius (p) on all the functions at 30D

Algorithm	LEP		FMP		SRP	
	DE	DE-CPI	DE	DE-CPI	DE	DE-CPI
DE/rand/1	**1.06e-009 5.85e-010**	3.41e-009 1.66e-008	**4.92e-001 2.46e+000**	3.53e+000 5.92e+000	2.44e+000 1.16e-001	**2.22e+000 4.20e-001**
DE/rand/2	3.36e-002 9.16e-003	**1.17e-005 5.22e-006**	1.38e+001 1.86e+000	**5.82e-001 2.67e+000**	2.46e+000 1.07e-001	**2.46e+000 7.71e-002**
DE/best/1	8.11e+000 6.39e+000	**5.42e+000 4.85e+000**	1.34e+001 5.39e+000	**1.14e+001 8.33e+000**	1.51e+000 1.98e-001	**1.47e+000 2.00e-001**
DE/best/2	0.00e+000 0.00e+000	0.00e+000 0.00e+000	3.61e+000 5.40e+000	**0.00e+000 0.00e+000**	2.45e+000 1.16e-001	**2.43e+000 1.43e-001**
DE/rand-to-best/1	8.12e-001 1.24e+000	**3.32e-004 1.19e-003**	1.41e+001 4.09e+000	**8.69e+000 7.54e+000**	1.76e+000 5.51e-001	**1.41e+000 1.91e-001**
DE/current-to-best/1	4.57e-001 1.72e+000	**0.00e+000 0.00e+000**	9.60e+000 6.64e+000	**4.31e+000 5.46e+000**	2.44e+000 1.32e-001	**2.14e+000 2.05e-001**
CoDE	2.32e+002 4.08e+001	**6.71e+001 1.45e+001**	5.53e+000 2.91e+000	**1.32e+000 1.81e+000**	**2.16e+000 8.30e-002**	2.17e+000 1.23e-001
JADE	0.00e+000 0.00e+000	0.00e+000 0.00e+000	**4.60e-001 2.30e+000**	1.34e+000 3.70e+000	1.49e+000 1.95e-001	**1.43e+000 1.72e-001**
jDE	6.93e-007 2.71e-006	**9.80e-008 3.01e-007**	2.60e-001 5.31e-001	**4.36e-018 2.18e-017**	1.99e+000 1.23e-001	**1.97e+000 9.25e-002**
MDE_pBX	**2.52e-003 6.30e-003**	9.04e-001 2.26e+000	**3.87e+000 6.63e+000**	5.85e+000 6.99e+000	2.00e+000 2.51e-001	**1.76e+000 2.00e-001**
ODE	6.52e-008 5.19e-008	**1.32e-008 1.29e-008**	**2.45e+000 4.96e+000**	2.89e+000 5.36e+000	**1.19e+000 1.89e-001**	1.26e+000 1.66e-001
SaDE	3.51e+001 1.11e+001	**5.16e+000 1.23e+000**	1.87e+000 1.75e+000	**6.49e-001 7.11e-001**	1.90e+000 8.01e-002	**1.88e+000 8.15e-002**

Table 7. Mean and standard deviation of the best error values obtained by the DE algorithms and DE-CPI for the real-world application problems

be quite difficult for the optimizers [36]. The *MNFES* for each problem is set to 150000, as suggested in [41]. The results are shown in Table 7.

From Table 7, we can find that DE-CPI can obtain the better solutions than the corresponding DE variants in most of the cases. Specifically, for SRP, DE-CPI is better than the corresponding DE variants in 10 out of 12 cases. DE-CPI is better than DE in 8 cases for FMP and 10 cases for LEP. In general, the results of Table 7 indicate that DE-CPI is able to improve the performance of DE effectively on the real-world problems considered.

6. Conclusion

In this study, a simple and effective DE framework, DE with composite population information based mutation operator (DE-CPI), is proposed to enhance the performance of DE for global numerical optimization. In DE-CPI, on the one hand, a generalized ring topology is used to define a neighborhood for each individual. On the other hand, the direction information is introduced into the mutation operator of DE by constructing the difference vector with the neighbors. In this way, the composite population information, i.e., neighborhood and direction information, can be fully and simultaneously utilized in DE-CPI to guide the search of DE. Through the extensive experimental study, DE-CPI is shown to be able to improve the performance of most of the DE algorithms studied.

In the future, the application of DE-CPI to other DE variants will be comprehensively studied firstly. Then, the adaptive or self-adaptive techniques for the neighborhood size will be investigated. Finally, the effectiveness of DE-CPI with other neighborhood topologies will also be studied.

7. Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (61305085, 61202468), the Natural Science Foundation of Fujian Province of China (2014J05074, 2014J01240), the Support Program for Innovative Team and Leading Talents of Huaqiao University (2014KJTD13) and the Fundamental Research Funds for the Central Universities (12BS216).

8. References

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

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

[3] Plagianakos, V., Tasoulis, D., Vrahatis, M. (2008). A review of major application areas of differential evolution, In: Advances in Differential Evolution, U. Chakraborty, Ed. Berlin, Germany: *Springer-Verlag*, 197-238.

[4] Wang, J., Cai, Y. (2014). Multi objective evolutionary algorithm for frequency assignment problem in satellite communications, *Soft Computing* (In press).

[5] Neri, F., Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis, *Artificial Intelligence Review*, 33 (1/2) 61-106, Feburary.

[6] Qin, A., Huang, V., Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation* t, 13 (2) 398-417.

[7] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation*, 10 (6) 646-657, December.

[8] Yu, W., Shen, M., Chen, W., Zhan, Z., Gong, Y., Lin, Y., Liu, O., Zhang, J. (2014). Differential evolution with two level parameter adaptation, *IEEE Transactions on Cybernetics*, (In press).

[9] Tang, L., Dong, Y., Liu, J. (2014). Differential Evolution with an Individual-Dependent Mechanism, *IEEE Transactions on Evolutionary Computation*, (In press).

[10] Zhang, J., Sanderson, A. (2009). JADE: adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation,* 13 (5) 945-958, October. 2009.

[11] Cai, Y., Wang, J. (2013). Differential evolution with neighborhood and direction information for numerical optimization, *IEEE Transactions on Cybernetics*, 43 (6) 2202-2215.

[12] Das, S., Abraham, A., Chakraborty, U. K., Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator, *IEEE Transactions on Evolutionary Computation*,13 (3) 526-553.

[13] Wang, J., Liao, J., Zhou, Y., Cai, Y. (2014). Differential evolution enhanced with multiobjective sorting based mutation operators, *IEEE Transactions on Cybernetics*, 46 (12) 2792-2805.

[14] Wang, Y., Cai, Z., Zhang, Q., Differential evolution with composite trial vector generation strategies and control parameters, *IEEE Transactions on Evolutionary Computation*, 15 (1) 55-66, Feburary.

[15] Cai, Y., Wang, Y. Chen, T. Wang, H. Tian and W. Luo. Adaptive direction information in differential evolution for numerical optimization, *Soft Computing*, 2014 (In press).

[16] Sun, J., Zhang, Q., Tsang, E. P. K. (2005). DE/EDA: A new evolutionary algorithm for global optimization, *Information Sciences*, 169 (3) 249-262.

[17] Dorronsoro, B., Bouvry, P. (2011). Improving classical and decentralized differential evolution with new mutation operator and population topologies, *IEEE Transactions on Evolutionary Computation*, 15, (1), p. 67-98, Feburary.

[18] Epitropakis, M. G., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., Vrahatis, M. N., Enhancing differential evolution utilizing proximity based mutation operators, *IEEE Transactions on Evolutionary Computation*,15 (1), 99 -119, Feburary.

[19] Weber, M., Neri, F., Tirronen, V. (2011). A study on scale factor in distributed differential evolution, I*nformation Sciences*, 181, (12) 2488-2511, June.

[20] Weber, M., Tirronen, V., Neri, F., Scale factor inheritance mechanism in distributed differential evolution, *Soft Computing*, 14 (11), 1187-1207, September.

[21] Neri, F., Iacca, G., Mininno, E. (2011). Disturbed exploitation compact differential evolution for limited memory optimization problems, *Information Sciences*, 181 (12) 2469-2487, June.

[22] Noman, N., Iba, H. (2011). Cellular differential evolution algorithm, *Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 293-302.

[23] Noroozi, V., Hashemi, A., Meybodi, M. (2011). CellularDE: A cellular based differential evolution for dynamic optimization problems, *Adaptive and natural computing algorithms*. Springer Berlin Heidelberg, 2011, 340-349.

[24] Dorronsoro, B., Bouvry, P. (2010). Differential evolution algorithms with cellular populations, *Parallel Problem Solving from Nature*. PPSN XI. *Springer Berlin Heidelberg*, 320-330.

[25] Omran, M., Engelbrecht, A., Salman, A. (2006). Using the ring neighborhood topology with self-adaptive differential evolution, *Advances in Natural Computation*, Eds. Berlin, Germany: Springer-Verlag, 976-979.

[26] Omran, M., Engelbrecht, A., Salman, A. (2009). Bare bones differential evolution, *European Journal of Operational Research*,196 (1) 128-139, July 2009.

[27] Cai, Y., Wang, J., Yin, J. (2012). Learning-enhanced differential evolution for numerical optimization, *Soft Computing*, 16, 2, 303 - 330, Feburuary.

[28] Fan, H., Lampinen, J. (2003). A trigonometric mutation operation to differential evolution, *Journal of global optimization*, 27 (1) 105-129, September.

[29] Wang, Y. X., Xiang, Q. L. (2008). Exploring new learning strategies in differential evolution algorithm, In: Proceedings *IEEE Congress on Evolutionary Computation*, 204 - 209.

[30] Bi, X. J., Xiao, J. (2011). Classification-based self-adaptive differential evolution with fast and reliable convergence performance, *Soft Computing*, 15 (8) 1581-1599, August.

[31] Iorio, A., Li, X. (2006). Incorporating directional information within a differential evolution algorithm for multi-objective optimization, *In*: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 691-698.

[32] Liu, J., Fan, Z., Goodman, E., SRDE: An improved differential evolution based on stochastic ranking, *In: Proceedings 1st ACM/SIGEVO*, 345-352.

[33] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Nanyang Technol. Univ., Singapore, KanGAL Rep. No. 2005005, May 2005, IIT Kanpur, India.

[34] Rahnamayan, S., Tizhoosh, H. R., Salama, M. M. A. (2008). Opposition based differential evolution, *IEEE Transactions on Evolutionary Computation*, 12 (1) 64-79.

[35] Islam, S. M., Das, S., Ghosh, S., Roy, S., Suganthan, P. N. (2012). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42 (2) 482-500, April.

[36] García, S., Fernández, A., Luengo, J., Herrera,F(2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability, *Soft Computing*, vol. 13, no. 10, p. 959 - 977.

[37] Derrac, J., García, S. Molina, D., Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary Computation*, 1 (1) 3-18.

[38] Alcala-Fdez, J., Sanchez, L., Garcia, S. KEEL: a software tool to assess evolutionary algorithms for data mining problems, [Online]. Available: http://www.keel.es/

[39] Zhan, Z. H., Zhang, J., Li, Y., Chung, H. S. H. (2009). Adaptive particle swarm optimization, *IEEE Transactions on Systems, Man, and Cybernetics,* 39 (6) 1362-1381, December.

[40] Eshelman, L. J., Mathias, K. E., Schaffer, J. D. (1997). Convergence controlled variation, In Foundations of Genetic Algorithms 4, R.Belew and M. Vose, Eds. SanMateo, CA, USA: *Morgan Kaufmann,* 203-224.

[41] Das, S., Suganthan, P. N. (2010). Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur Univ., West Bengal, India, Tech. Rep., December 2010, Nanyang Technological University, Singapore.

9. Author biographies

Jingliang Liao received the B.S. degree from Quanzhou Normal University, Quanzhou, China, in 2012. He is currently working toward the M.S. degree in the College

of Computer Science and Technology, Huaqiao University, Xiamen, China. His research interests are differential evolution, evolutionary computation and machine learning.

Yiqiao Cai is currently a Lecturer with the College of Computer Science and Technology at Huaqiao University. He received the B.S. degree from Hunan University, Changsha, China, in 2007 and the Ph.D. degree from Sun Yatsen University, Guangzhou, China, in 2012. His research interests are differential evolution, multiobjective optimization, and other evolutionary computation techniques.

Yonghong Chen is now a professor of College of Computer Science and Technology at Huaqiao University. He received the Ph.D. degree from College of Automation at Chongqing University in 2005. His research interests include network and information security, cloud computing, and Internet of things technology.

Tian Wang is currently a Lecturer with the College of Computer Science and Technology at Huaqiao University. He received the B.Sc. and M.Sc. degrees in computer science from the Central South University, Changsha, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from the City University of Hong Kong, Kowloon, Hong Kong, SAR, in 2011. His research interests include wireless sensor networks, Internet of things, and mobility-based video surveillance and tracking.

Hui Tian is now an associate professor of College of Computer Science and Technology at Huaqiao University. He received the B.Sc. degree and the M.Sc degree in 2004 and 2007 from Wuhan Institute of Technology, Wuhan, China, and the PhD degree in 2010 from Huazhong University of Science and Technology, Wuhan, China. His present research interests include network and multimedia information security, digital forensics, information hiding and covert communication, and intelligent computing.