Journal of Computers Vol. 30 No. 5, 2019, pp. 46-59
doi:10.3966/199115992019103005004

Integration Differential Evolution Algorithm with Dynamic Multiple Population Base on Weighted Strategies

Yu-Ling Fan[1], Jia-Neng Tang[1,2,3*], Pei-Zhong Liu[1], Yan-Ming Luo[4], Xiao-Fang Liu[1]

[1] College of Engineering, Huaqiao University, Quanzhou, Fujian, China

[2] College of mechanical engineering and automation, Huaqiao University, Xiamen, Fujian, China

[3] Centron Communications Technologies Fujian Co., Ltd., Quanzhou, Fujian, China

tangjianeng@sina.com

[4] College of Computer Science and Technology, Huaqiao University, Xiamen, Fujian, China

Received 26 October 2017; Revised 28 February 2018; Accepted 31 March 2018

Abstract. Integration differential evolution algorithm with dynamic multiple population base on weighted strategies (MPWDE) is a kind of heuristic random search algorithm, which is used to solve global optimization problems. Firstly, the population is divided into several sub populations, which improves the diversity of the population at the initial stage of the population. Meanwhile, the population diversity is improved by the parabola increment crossover factor in evolutionary process. Then, by introducing weighted strategies mechanism, the mutation strategy "DE/current-to-pbest/2 or DE/current-to-rbest/2" improves the convergence speed of the algorithm. Finally, the numerical simulation results show the effectiveness of the proposed algorithm, compared with the state-of-the-art algorithms on CEC2006 benchmark functions.

Keywords: differential evolution, multiple populations, parabola increment, weighted strategy

1 Introduction

Differential evolution algorithm (DE) is a heuristic random search algorithm that simulates the difference of population evolution [1]. DE were originally conceived to solve the Chebyshev polynomial problem, but later Storn and Price found that compared with other evolutionary algorithms, it was more prominent in solving complex global optimization problems. The evolutionary process is simpler, less controlled parameters and strong adaptability. At present, it has been widely used in solving practical optimization problems, such as process simulation, engineering design optimization, economic and environmental distribution optimization [2].

In addition, the performance of the DE is highly dependent on mutation strategies and control parameters, such as the population size *NP*, scaling factor *F* and crossover factor *CR*. In general, when DE solves different optimization problems, the most appropriate variation strategy and control parameters can be different. Even for a particular optimization problem, the best strategy and control parameters may be different during the evolution. Especially, it is valid to solve all kinds of optimization problems by using the traditional iterative experiment to determine the best strategy and parameters, but it consumes time.

In order to solve the above problems, international and domestic academics have made many improvements about the standard DE, which mainly includes the following aspects: the adaption of control parameters, the improvement of mutation strategy, the use of multiple mutation strategies and the preservation of population diversity from the improved algorithm. Many enhanced DE variants such as jDE (with self-adapted parameters) [3], SaDE (with adapted mutation strategies and parameters) [4], CoDE (composition of multiple strategies and parameter settings) [5], JADE (with "current-to-pbest/1"

[*] Corresponding Author

mutation strategy and the adaptive parameter) [6], DE-DPS (dynamically selecting the best combination of parameters) [7], CoBiDE (covariance matrix learning and bimodal distribution parameter) [8], EPSDE (with ensemble of mutation strategies and parameters) [9], LSAOS-DE (landscape-based adaptive operator selection mechanism) [10], ARDE (a hybrid of four commonly used repair methods) [11], BetaCODE (stochastic opposition-based learning using a beta distribution) [12], LS-EXP (a linearly scalable exponential crossover operator) [13], FA (a hybrid fireworks algorithm) [14], APTSDE (adaptive population tuning scheme for differential evolution) [15], Rank (differential evolution with rank-based mutation operators) [16], DEGL/SAW (differential evolution using a Neighborhood-Based Mutation Operator) [17], self-CCDE and self-CSDE (with cluster-based strategy and self-adaptive parameter control) [18], MPEDE (Multi-Population Based Ensemble of Mutation Strategies) [19], have been proposed.

Compared with previous works, the main contributions are proposed to address the exploration and exploitation trade-off issue: Dynamic multiple population approach is the initial division of population and the redistribution of subpopulation in the evolution process. The method is presented for improving the population diversity. Most state-of-the-art DE variants (JADE, jDE, SaDE and EPSDE) use single population. Subsequently, weighted mutation strategy is incorporated to DE to avoid stagnant and precocious phenomenon caused by MPEDE. In addition, the ability of reconnaissance and convergence is balanced by using a parabola type of adaptive control parameters. MPWDE is tested on the suit of CEC2006 benchmark functions with 30 and 50 variables, respectively. The competitive performance of MPWDE is exhibited by extensive comparisons with several state-of-the-art DE variants.

The remainder of this paper is organized as follows. Section 2 briefly introduces DE and its operators. Then, MPWDE is presented in Section 3. The experimental results are given in Section 4. Section 5 concludes this paper.

2 Differential Evolution

The differential evolution algorithm can be thought of a kind of greedy genetic algorithm which is based on real number coding. In the evolutionary stage, the individual population enters the iteration process through variation, crossover and selection until the stopping condition is satisfied. Each individual in the population represents a candidate solution of the objective function *f(x)*, which evaluates its quality by calculating the adaptive function and records the optimal individual.

2.1 Population Initialization

The population is initialized as follows: *D* represents the dimension of feasible solution space, X^G represents the *G* generation population (i.e., $X^G = \{x_1^G, x_2^G, ..., x_{NP}^G\}$). Each individual $x_{i,j}^G$ is made up of *D* dimensional parameters:

$$X_i^G = [x_{i,1}^G, x_{i,2}^G, ..., x_{i,D}^G], i \in \{1, 2, ..., NP\}. \quad \textbf{(1)}$$

Where $x_{i,j}^G$ is randomly chosen within the range $[X^L, X^H]$, X^L represents the lower bound, X^H represents the upper bound.

2.2 Mutation Operator

The commonly used mutation operator can be formulated as follows:

$$v_i^{G+1} = x_{\gamma 1}^G + F \cdot (x_{\gamma 2}^G - x_{\gamma 3}^G). \quad \textbf{(2)}$$

At this stage, the parent population x_i^G produce variant individual v_i^{G+1} by mutation strategies, $F \in [0, 1]$. The indices *r1*, *r2*, *r3* should be mutually exclusive and are generated randomly once every mutant vector within the range of $[1, NP]$.

2.3 Crossover Operator

The main function of cross operation is to generate the trail vector u_i by the target vector x_i and the mutated vector v_i. The DE algorithm adopts a binomial cross scheme, and the crossover operation can be formulated as follows:

$$u_{i,j}^{G+1} = \begin{cases} v_{i,j}^{G+1}, if\ and_j \le CR\ or\ j = j_{rand} \\ x_{i,j}^{G}, \qquad otherwise \end{cases}. \tag{3}$$

Where j_{rand} is a random integer between 1 and D, the arbitrary dimension from the mutation vector to the target vector. $rand_j$ is a uniformly distributed random number between 0 and 1, and CR is the crossover control parameter.

2.4 Selection Operator

The selection operation mainly adopts the greedy selection model of the survival of the fittest. Then the offspring are always superior to or equal to the parent individual. The trail vector u_i is accepted by the population if and only if the fitness value of u_i is better than the fitness value of the target vector x_i. Otherwise, x_i remains in the next generation:

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, if\ f(u_i^{G+1}) \le f(x_i^G) \\ x_i^G, \qquad otherwise \end{cases}. \tag{4}$$

Through the above principle, we can see that the standard differential evolution algorithm is easy to implement. In terms of solving the specific problems to other industry experts, DE is easier to apply to this situation. Although the DE algorithm has many advantages, there is still a single variation strategy, control parameters fixed and so on. In order to avoid the difficulties caused by the above problems, this paper proposes MPWDE.

3 Dynamic Multiple Population Base on Weighted Strategies

3.1 Dynamic Multiple Population Approach

In the evolutionary process with the increase of the number of iterations, the individual differences gradually decrease. An aggregation phenomenon is formed. Then this phenomenon leads to premature convergence in the search global optimization, leaves the algorithm into local optimum. The idea of dividing the whole population into subpopulation is introduced. The number of individuals of the next generation of subpopulation is determined by the evolutionary results of each iteration subpopulation. Pop is the whole population, Pop_i is the subpopulation of Pop. The subpopulation Pop_i consist of population size NP_i and population scale ratio λ_i. The dynamic multiple population approach is more able to ensure the diversity of the population and avoid the singularity of the population, which can be formulated as follows:

$$\begin{cases} Pop = \bigcup_{i=1,\dots,n} Pop_i \\ NP_i = \lambda_i * NP \\ \bigcup_{i=1,\dots,n} NP_i = NP \end{cases}. \tag{5}$$

In this paper, we divide the population into three subpopulations, say Pop_1, Pop_2 and Pop_3 fixedly every generation. Pop_2 and Pop_3 have the same size while the size of Pop_1 is much larger than these two subpopulations. In this study, we let $\lambda_1 > \lambda_1 = \lambda_3$, λ_i is chosen from [0, 1].

At first, subpopulation Pop_1, Pop_2 and Pop_3 are assigned to the corresponding mutation strategies. Along with evolution, every edp_{mum_i} (the reservation of fine individual) number of generations is counted in subpopulation. From the edp_{mum_i}, the determined best performed mutation strategy will be given the largest population size. After each population evolution, the idea of reward subpopulation assignment operations shows above, then the subpopulation excellent rate nr_i can be formulated as follows:

$$nr_i = edp\_num_i / NP_i, i = 1, 2, 3. \tag{6}$$

The largest population gives more resources to the best performed mutation strategy which dominates the optimization process. For the multiple population mechanism, each subpopulation can share optimization experience with each other and exchange information at every generation.

3.2 Weighted Mutation Strategy

Mutation strategy is the core of DE algorithm. Due to the different mutation mechanism, there is a great diversity mutation strategy. To distinguish the different mutation strategy, the strategy is generally expressed as DE/*X*/*Y*/*Z*. Obviously, *X* represents the base vector to be perturbed, *Y* represents the number of differential vectors, *Z* represents crossover mode. In this paper, three subpopulations use different strategies to evaluate the diversity of individual populations. Learn from the experience of the past improved classic DE algorithm, three strategies are chosen in paper.

(1) "DE/current-to-best /1"

$$v_i^{G+1} = X_i^G + F \cdot (X_{best}^G - v_{\gamma 1}^G). \tag{7}$$

(2) "DE/current-to-rand/1"

$$v_i^{G+1} = X_i^G + K \cdot (X_{\gamma 1}^G - X_i^G) + F \cdot (X_{\gamma 2}^G - X_{\gamma 3}^G). \tag{8}$$

(3) "DE/current-to-best/2"

$$v_i^{G+1} = X_i^G + F \cdot (X_{best}^G - X_i^G) + F \cdot (X_{\gamma 1}^G - X_{\gamma 2}^G). \tag{9}$$

All the mutation vectors in Eq. (8) are chosen in a random way, so they can be optimized in the global. At the same time, the convergence rate of Eq. (8) is slower due to the lack of optimal solution information. In contrast, Eq. (7) and Eq. (9) can find the global optimal solution in the current generation. In the evolutionary process, Eq. (7) and Eq. (9) can narrow the search range near the optimal solution in a quick speed of convergence, but the population diversity is missing and the optimal solution cannot be found. In order to reduce the influence of the narrow research in earlier stage and the slow convergence rate in the later period, the mutation strategy of Eq. (9) is modified to establish the weighted strategy mechanism. The combination of the two mutation strategies is established for one subpopulation (local strategy L_i^{G+1} and global strategy Q_i^{G+1}). is the best individual in the subpopulation, X_{best}^G is the best individual in the whole population.

Local variation:

$$L_i^{G+1} = X_i^G + F \cdot (X_{\gamma best}^G - X_i^G) + F \cdot (X_{\gamma 1}^G - X_{\gamma 2}^G). \tag{10}$$

Global variation:

$$Q_i^{G+1} = X_i^G + F \cdot (X_{best}^G - X_i^G) + F \cdot (X_{\gamma 1}^G - X_{\gamma 2}^G). \tag{11}$$

The weighted mutation strategy uses the linear variable *w* as a weight to balance the local and global mutation strategy. w_{min} and w_{max} are weighted factor *w* upper and lower bounds. *t* indicates the number of generation and is *FES* the number of maximum function evaluations.

$$\begin{cases} v_i^{G+1} = (1-w)L_i^{G+1} + w \cdot Q_i^{G+1} \\ w = w_{\min} + (w_{\max} - w_{\min}) \cdot t / FES \end{cases}. \quad \textbf{(12)}$$

At the start of the algorithm, $t = 0$, $w = w_{min}$. As the number of iterations increases, w gradually ascend. When t is equal to FES, w reaches w_{max}. The algorithm develops from the local search patterns to global patterns. Obviously, in order to achieve the balance between local variation and global variation, it is necessary to select $w_{\min}$ and $w_{\max}$ appropriately.

3.3 Parameter Adaptation

The algorithm need adopt the global mutation strategy in the early stage and the local mutation strategy in the later period for a quick speed of the convergence rate. In this paper, the F is adopted in Cauchy distribution. c is adaptive parameters to control the speed, γ is the scale parameter of Cauchy distribution. The previous generation excellent scaling factor is placed on the *goodF*. The operation can be formulated as follows:

$$\begin{cases} F_m = (1-c) * \mu F + c * goodF \\ F = F_m + \gamma * \tan(\pi(rand - 0.5)) \end{cases}. \quad \textbf{(13)}$$

The initial value of μF is 0.5. In the evolution process, the value of the F determines the mutation range of the basis vector. The larger the F value, the larger the mutation range and vice versa.

In the classical DE, the value of the CR is a constant. CR determines that the probability of the new individuals will inherit genes from v_i^{G+1}. The larger the CR value, the greater the contribution of mutation individuals. *gen* is the generation of current evolution, *Gen* is the final evolutionary generation of population. CR_{min} and CR_{max} indicate the lower bound and the upper bound. In order to improve the performance of the algorithm, a parabola type of the dynamic increasing CR is presented in Eq.(14).

$$CR = CR_{\min} + (CR_{\max} - CR_{\min}) \cdot gen \wedge 2 / Gen \wedge 2 \quad \textbf{(14)}$$

This strategy aims at changing the value of CR in the evolution process. The effects of the linear parameters and the fixed parameters are reduced in the algorithm.

3.4 Framework of MPWDE

MPWDE combines the dynamic multiple population mechanism with the weighted mutation strategy, then the pseudocode of MPWDE is shown in Table 1.

Table 1. The pseudocode of MPWDE

```
Input: $NP$: the number of individuals contained by the population
       $FES$: maximum number of function evaluations
(1)  $gen$=0;
(2)  Generate an initial population $X^G = \{x_1^G, x_2^G, ..., x_{NP}^G\}$ by randomly sampling from the search space;
(3)  Evaluate the objective function values of each individual (i.e., each target vector) in $X^G$;
(4)  $t = NP$; /*t records the number of function evaluations */
(5)  Generate the initial scaling factor $F_{i,0}$ and crossover control parameter $CR_{i,0}$ (i.e., $i \in \{1,\dots,NP\}$) for each
     target vector $X^0$ the population according to Eq.(13) and Eq.(14), respectively;
(6)  While $t < FES$
(7)      $Pop_1 \neq \phi, Pop_2 \neq \phi, Pop_3 \neq \phi$;
(8)      For $i$=1: $NP_1$
(9)      Apply the mutation operator (i.e., Eq.(7)) to produce a mutant vector $v_i^{G+1}$;
(10)         Implement the crossover operator (i.e., Eq.(4)) produce a trial vector $u_i^{G+1}$;
(11)         Evaluate the objective function value of $u_i^{G+1}$;
(12)         IF $f(u_i^{G+1}) \leq f(x_i^{G+1})$
```

```
(13)         $X^{G+1} = X^{G} \cup u_i^{G+1}$;
(14)         $F_{i,G+1} = F_{i,G}$ and $CR_{i,G+1}$ according to Eq.(14);
(15)      Else
(16)         $X^{G+1} = X^{G} \cup X_i^{G+1}$;
(17)         $F_{i,G+1}$ and $CR_{i,G+1}$ according to Eq.(13) and Eq.(14);
(18)      End IF
(19)    End For
(20)    For $i$ =1: $NP_2$
(21)      Apply the mutation operator (i.e., Eq.(8)) to produce a mutant vector $v_i^{G+1}$;
(22)      Implement the crossover operator (i.e., Eq.(4)) produce a trial vector $u_i^{G+1}$;
(23)      Evaluate the objective function value of $u_i^{G+1}$;
(24)      IF $f(u_i^{G+1}) \le f(x_i^{G})$
(25)         $X^{G+1} = X^{G} \cup u_i^{G+1}$;
(26)         $F_{i,G+1} = F_{i,G}$ and $CR_{i,G+1}$ according to Eq.(14);
(27)      Else
(28)         $X^{G+1} = X^{G} \cup X_i^{G}$;
(29)         $F_{i,G+1}$ and $CR_{i,G+1}$ according to Eq.(13) and Eq.(14);
(30)      End IF
(31)    End For
(32)    For $i$ =1: $NP_3$
(33)      Apply the mutation operator (i.e., Eq.(12)) to produce a mutant vector $v_i^{G+1}$;
(34)      Implement the crossover operator (i.e., Eq.(4)) produce a trial vector $u_i^{G+1}$;
(35)      Evaluate the objective function value of $u_i^{G+1}$;
(36)      IF $f(u_i^{G+1}) \le f(x_i^{G})$
(37)         $X^{G+1} = X^{G} \cup u_i^{G+1}$;
(38)         $F_{i,G+1} = F_{i,G}$ and $CR_{i,G+1}$ according to Eq.(14);
(39)      Else
(40)         $X^{G+1} = X^{G} \cup X_i^{G}$;
(41)         $F_{i,G+1}$ and $CR_{i,G+1}$ according to Eq.(13) and Eq.(14);
(42)      End IF
(43)     End For
(44)     $t = t + NP$;
(45)     $gen = gen + 1$;
(46) End While
Output: the individual with the smallest objective function value in the population
```

4 Experimental Study

4.1 Experiment Setting

In order to verify the performance of the MPWDE algorithm proposed in this paper, the algorithm is tested by 14 test functions in the benchmark function. The benchmark function includes the unimodal function (f_1–f_5), the basic multimodal function (f_6–f_{12}) and the extended multimodal function (f_{13}–f_{14}). For more information, please refer to the article [20]. MPWDE was compared with five other state-of-the-art DE variants including jDE, JADE, SaDE, EPSDE, MPEDE. The reason for choosing the above DE algorithm is as follows: First, JADE and jDE are very efficient and frequently cited in literature as baseline algorithms. Second, SaDE and EPSDE use multiple strategies as mutation strategy. Third MPEDE is the best performing algorithm for global optimization problems.

To provide a more comprehensive comparison, we run each comparative algorithm 25 times over the benchmark functions with 30 and 50 decision variables. The allowed *FES* for the benchmark functions with 30 and 50 decision variables are set to 300 000 and 500 000, respectively. Experimental parameter settings: *NP*=250. Scaling ratio λ_1 =0.2, λ_2 =0.2, λ_3 =0.6. Cross factor CR_{min}=0.5, CR_{max}=0.9. The experimental environment is as follows: Operating system is windows7 professional 64-bit, CPU (Central processing Unit) is core i7 (3.40GHz), RAM (random access memory) is 8GB, Language is matlab, Compiler is MATLAB R2014b.

4.2 Comparison with Other State-of-the-Art EAs

DE algorithms compared under this subsection have three aspects. Firstly, population structure plays an important role in population diversity. Secondly, instead of using a single offspring generation strategy, they select one such strategy from a pool of a few possible strategies. Thirdly, they include some mechanisms for adapting parameter values also. The following introduces the difference between six of DE algorithms in Table 2.

Table 2. The difference between six of DE algorithms

Function	Population Structure	Mutation Strategy	Parameter(*CR*)
JADE	single population	single strategy(current-to-best)	normal distribution
jDE	single population	single strategy(rand)	random distribution
SaDE	single population	strategy pool	normal distribution
EPSDE	single population	strategy pool	parameter pool
MPEDE	two type of subpopulations	single strategy(current-to-pbest)	normal distribution
MPWDE	three subpopulations	weighted strategy	parabola type

This information can be derived from the Table 2, we point out the differences between the compared algorithms as follows: JADE, jDE, SaDE, and EPSDE use the single population, two type of subpopulation is employed in MPEDE. The population of the MPWDE is divided into three sub-populations and reassign in next generation for improving the population diversity. JADE, jDE and MPEDE use single strategy, SaDE and EPSDE select one strategy from the strategy pool. Weighted strategy, “current-to-pbest/1” and “current-to-rand/2” are taken as mutation strategies to avoid stagnant and precocious phenomenon in MPWDE. The four compared algorithms use normal distribution on the parameter, a parabola type of the *CR* is more stable than the other algorithms.

The computational results obtained by running each of the six comparative DE variants 25 times on each benchmark function with 30 and 50 variables are reported in Table 3 and Table 4, respectively. The mean error and standard deviation (in bracket) of the function error values are provided in the two tables. Results obtained by MPWDE are highlighted if they are the best. In addition, Wilcoxon’s rank sum test at a 0.05 significance level is conducted between MPWDE and JADE, jDE, SaDE, EPSDE and MPEDE. Signs “−”, “+”, and “≈” indicate that the related comparative DE variant is significantly worse than, better than, and similar to MPWDE, respectively.

Table 3. Computational result of benchmark functions with 30 variables

Function	JADE	jDE	SaDE	EPSDE	MPEDE	MPWDE
f_1	0.00E+00 (0.00E+00)	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)
f_2	1.93e-28 (1.48e-28) −	1.24e-06 (1.46e-06) −	1.40e-05 (1.83e-05) −	1.32e-24 (6.56e-24) −	1.04E-26 (4.46E-26) −	4.23e-29 (5.52e-29)
f_3	1.43e+04 (1.18e+04) −	1.83e+05 (1.22e+05) −	3.94e+05 (2.03e+05) −	1.08e+06 (4.49e+06) −	1.41E+01 (5.29E+01) −	1.03e-05 (5.15e-05)
f_4	1.46e-10 (2.91e-10) −	3.41e-02 (7.77e-02) −	1.86e+01 (3.67e+01) −	2.07e-03 (1.01e-02) −	2.19e-17 (9.93e-17) −	2.06e-23 (6.58e-23)
f_5	1.86e-02 (8.21e-02) −	5.49e+02 (4.77e+02) −	2.44e+03 (5.52e+02) −	7.27e+02 (4.45e+02) −	9.55e-06 (2.00e-06) −	8.98e-09 (2.88e-08)
f_6	1.01e+01 (2.42e+01) −	2.52e+01 (2.56e+01) −	4.81e+01 (3.02e+01) −	2.87e+00 (1.38e+01) −	2.37e+00 (5.05e+00) −	1.83e-14 (9.17e-14)

Table 3. Computational result of benchmark functions with 30 variables (continue)

Function	JADE	jDE	SaDE	EPSDE	MPEDE	MPWDE
f_7	4.69e+03 (2.38e-12) −	4.69e+03 (2.63e-12) −	4.69e+03 (9.28e-13) −	1.86e+00 (1.69e+00) −	5.22e-03 (6.86e-03) −	3.05e-03 (5.19e-03)
f_8	2.09e+01 (5.29e-02)≈	2.09e+01 (5.31e-02)≈	2.09e+01 (5.79e-02)≈	2.09e+01 (5.55e-02)≈	2.09e+01 (4.94e-01)≈	2.09e+01 (5.39e-02)
f_9	0.00e+00 (0.00e+00)+	0.00e+00 (0.00e+00)+	0.00e+00 (0.00e+00)+	0.00e+00 (0.00e+00)+	0.00e+00 (0.00e+00)+	3.79e-13 (4.93e-13)
f_{10}	2.55e+01 (4.89e+00)+	5.19e+01 (8.11e+00) −	4.08e+01 (1.27e+01) −	5.63e+01 (8.35e+00) −	2.35e+01 (7.22e+00)+	3.06e+01 (1.15e+01)
f_{11}	2.53e+01 (1.52e+00) −	2.78e+01 (1.72e+00) −	2.27e+01 (7.79e+00) −	3.48e+01 (3.24e+00) −	2.15e+01 (6.91e+00) −	1.69e+01 (5.68e+00)
f_{12}	6.11e+03 (6.01e+03) −	4.36e+03 (6.21e+03) −	6.46e+03 (8.23e+03) −	5.83e+04 (8.91e+03) −	2.03e+03 (2.11e+03)+	3.67e+03 (5.48e+03)
f_{13}	1.46e+00 (9.08e-02)+	1.72e+00 (1.34e-01)+	4.64e+00 (4.30e-01) −	2.54e+00 (2.47e-01) −	2.92e+00 (6.33e-01) −	1.94e+00 (2.86e-01)
f_{14}	1.23e+01 (2.79e-01)≈	1.30e+01 (2.32e-01)≈	1.28e+01 (1.93e-01) −	1.35e+01 (1.75e-01) −	1.24e+01 (2.64e-01)≈	1.22e+01 (4.37e-01)
−/+/≈	9/2/3	8/3/3	11/1/2	11/1/2	8/3/3	

From the data given in Table 3, we can make several observations and conclusions. First, for Unimodal functions (f_1–f_5), JADE and MPEDE algorithms have good experimental results. In fact, MPWDE experimental results better, the results are more significant differences; For three unimodal functions (f_3, f_4, f_5), MPWDE experimental results are significantly better than JADE and MPEDE algorithm; For two unimodal functions (f_1, f_2), MPWDE is comparable to the contrast algorithm. Second, for basic multi-modal benchmark functions (f_6–f_{12}), the effect of MPWDE and MPEDE are more obvious than the rest algorithms. The result of MPWDE is better than MPEDE on two benchmark functions (f_6, f_{11}). Third, the experimental results of the six algorithms for the extended multimodal function (f_{13}–f_{14}) are similar. Finally, we can conclude that the experimental results of MPWDE algorithm on benchmark functions with 30 variables are better than jDE, JADE, SaDE, EPSDE and MPEDE. Actually, the results of Wilcoxon's rank sum tests reported in the last three rows indicate that MPWDE is significantly better than JADE, jDE, SaDE, EPSDE and MPEDE on 9, 8, 11, 11 and 8 functions, respectively. It is significantly worse than JADE, jDE, SaDE, EPSDE and MPEDE on 2, 3, 1, 1 and 3 functions and similar to them on 3, 3, 2, 2 and 3 functions, respectively.

Table 4. Computational result of benchmark functions with 50 variables

Function	JADE	jDE	SaDE	EPSDE	MPEDE	MPWDE
f_1	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)
f_2	1.71e-21 (4.04e-21)+	1.88e-02 (2.85e-02) −	1.07e-01 (9.01e-02) −	3.32e+02 (1.65e+03) −	7.81e-13 (1.54e-12) −	8.92e-17 (1.51e-16)
f_3	2.35e+04 (1.22e+04)+	4.81e+05 (2.23e+05) −	9.02e+05 (2.61e+05) −	1.26e+07 (3.44e+07) −	6.46e+04 (2.91e+04) −	3.61e+04 (2.08e+04)
f_4	1.01e+01 (1.73e+01) −	3.71e+02 (4.46e+02) −	3.03e+03 (1.32e+03) −	1.97e+02 (3.45e+02) −	7.60e+00 (3.43e+01) −	4.72e-02 (1.51e-02)
f_5	2.08e+03 (3.64e+02) −	3.25e+03 (5.62e+02) −	5.87e+03 (9.76e+02) −	2.55e+03 (6.31e+02) −	5.67e+02 (4.56e+02) −	1.71e+02 (2.96e+02)
f_6	1.42e+01 (4.31e+00) −	4.22e+02 (3.02e+01) −	9.20e+01 (3.84e+01) −	3.58e+00 (1.54e+01) −	1.18e+00 (1.82e+00) −	4.78e-01 (1.32e+00)
f_7	6.19e+03 (1.84e-12) −	6.19e+03 (3.28e-12) −	6.19e+03 (7.87e-13) −	1.00e+00 (1.21e-01) −	4.04e-03 (7.24e-03)+	7.58e-03 (9.28e-03)
f_8	2.11e+01 (2.29e-02)≈	2.11e+01 (3.02e-02)≈	2.11e+01 (4.46e-02)≈	2.11e+01 (3.63e-02)≈	2.11e+01 (3.87e-02)≈	2.11e+01 (3.28e-02)
f_9	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)≈	0.00e+00 (0.00e+00)
f_{10}	5.42e+01 (1.02e+01)+	9.87e+01 (1.56e+01) −	1.01e+02 (1.77e+01) −	1.89e+02 (2.91e+01) −	4.75e+01 (1.22e+01)+	6.14e+01 (1.57e+01)

Table 4. Computational result of benchmark functions with 50 variables (continue)

Function	JADE	jDE	SaDE	EPSDE	MPEDE	MPWDE
f_{11}	5.24e+01 (1.92e+00) −	5.46e+01 (2.21e+00) −	3.51e+01 (7.73e+00) −	6.99e+01 (3.61e+00) −	3.59e+01 (1.22e+01) −	3.35e+01 (1.08e+01)
f_{12}	1.49e+04 (1.11e+04) −	2.15e+04 (2.24e+04) −	1.62e+04 (8.58e+03)+	4.55e+05 (5.16e+04) −	8.55e+03 (6.13e+03)+	1.64e+04 (1.93e+04)
f_{13}	3.83e+00 (1.49e-01) −	4.01e+00 (2.13e-01) −	1.06e+01 (5.73e-01) −	7.80e+00 (5.18e-01) −	4.15e+00 (4.12e-01) −	2.80e+00 (4.97e-01)
f_{14}	2.17e+01 (4.27e-01) −	2.26e+01 (3.05e-01) −	2.25e+01 (1.91e-01) −	2.34e+01 (2.02e-01) −	2.17e+01 (4.08e-01) −	2.12e+01 (4.57e-01)
−/+/≈	11/0/3	8/3/3	10/1/3	11/0/3	8/3/3	

From the data given in Table 4, we can make several observations and conclusions. First, for Unimodal functions (f_1–f_5), the experimental result of algorithm is similar in f_1. The experimental results of JADE and MPEDE are a little bit better than MPWDE in two unimodal functions (f_2, f_3). The experimental results of MPWDE in f_4 and f_5 are superior to all contrast algorithms. Second, for basic multi-modal benchmark functions (f_6–f_{12}), MPWDE and MPEDE are more effective than the other contrast algorithms. Especially on two basic multi-modal benchmark functions (f_6, f_{11}), the MPWDE results are better than the MPEDE, and the rest of the benchmark functions have similar testing results. Third, for the extended multimodal benchmark functions (f_{13}–f_{14}), the test results of MPWDE are more obvious than that of all the contrast algorithms. Finally, we can conclude that the experimental results of MPWDE algorithm on benchmark functions with 50 variables are better than jDE, JADE, SaDE, EPSDE and MPEDE. Actually, the results of Wilcoxon's rank sum tests reported in the last three rows indicate that MPWDE is significantly better than JADE, jDE, SaDE, EPSDE and MPEDE on 11, 8, 10, 11 and 8 functions, respectively. It is significantly worse than JADE, jDE, SaDE, EPSDE and MPEDE on 0, 3, 1, 0and 3 functions and similar to them on 3, 3, 3, 3 and 3 functions, respectively.

The performance of the algorithm in the data of Table 3 and Table 4 is evaluated by evolution curve. The evolution curve takes Average Function Value as the vertical axis, and the evaluation number *FES* is the horizontal axis. The detailed analysis is as follows:

The comparison between five state-of-the-art DE algorithms and MPWDE shows that MPWDE has strong advantages both in convergence speed and in convergence performance. From the unimodal functions (e.g. Fig. 1), jDE and SaDE have a single strategy, which results in a relatively small variation of population variation and early maturity. No convergence trend appears in ESPDE, JADE and MPEDE, but the accuracy of the solution is less than that of MPWDE. From unimodal function (e.g. Fig. 4), jDE, SaDE and ESPDE are precocious. Although JADE and MPEDE may have the opportunity to find the global optimal solution in the iteration, the accuracy is low and the convergence rate is far less than that of MPWDE. For the basic multimodal benchmark function (e.g. Fig. 5), five state-of-the-art DE algorithms show premature convergence. It can be seen from the figure that MPWDE can avoid entering the local extreme point and quickly search the global optimal solution. It can be seen from the graphs (e.g. Fig. 8, Fig. 9, Fig. 10 and Fig. 12) that the convergence rates of the six algorithms are basically the same. The population diversity in the early stage of MPWDE is large, resulting in a higher accuracy than the other algorithms in later. From the picture (e.g. Fig. 5, Fig. 6, Fig. 7, Fig. 11 and Fig. 13), we can see that the convergence of the MPWDE algorithm is slow in the global search, and then gradually search from the global transition to improve the accuracy of the solution. Not only that, from unimodal function (e.g. Fig. 2, Fig. 3 and Fig. 4), it can be clearly seen MPWDE convergence speed and solution accuracy are optimal. Population diversity can be seen from the picture (e.g. Fig. 1 to Fig. 13), when the population lost diversity, the algorithm will appear precocious. Although the population diversity and convergence rate are contradictory, this algorithm not only controls the diversity of population, but also applies the weighting strategy to make the search from global to local, so the algorithm is superior to the other five algorithms in solving accuracy and convergence speed.

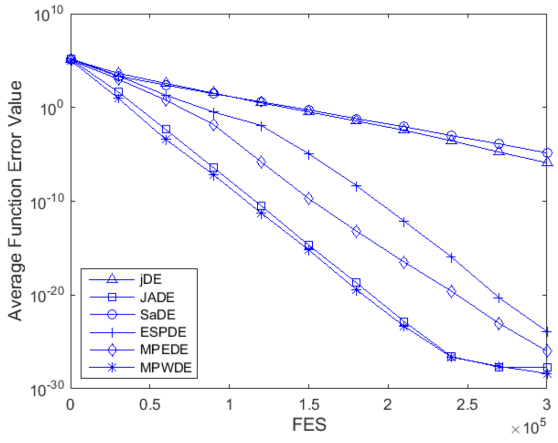

Fig. 1. 30 dimensional f_2 function

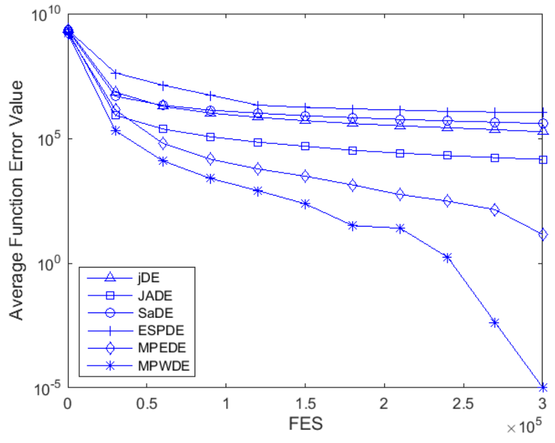

Fig. 2. 30 dimensional f_3 function

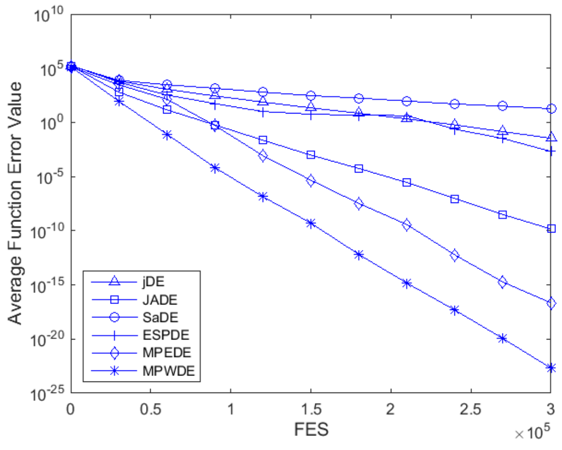

Fig. 3. 30 dimensional f_4 function

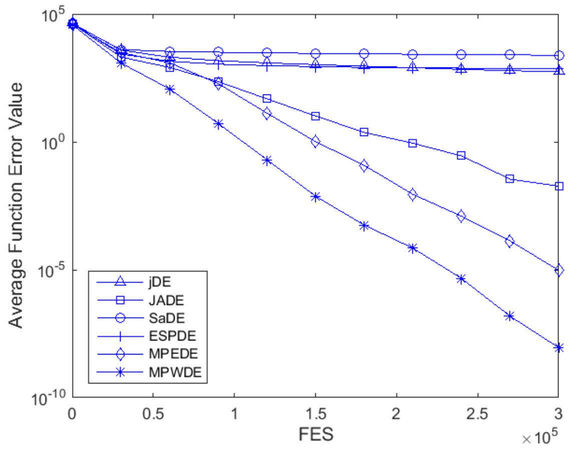

Fig. 4. 30 dimensional f_5 function

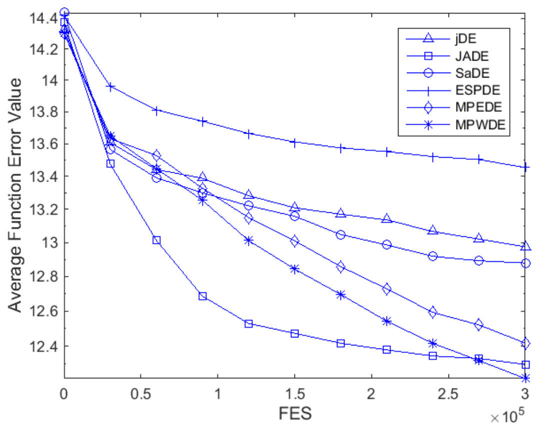

Fig. 5. 30 dimensional f_6 function

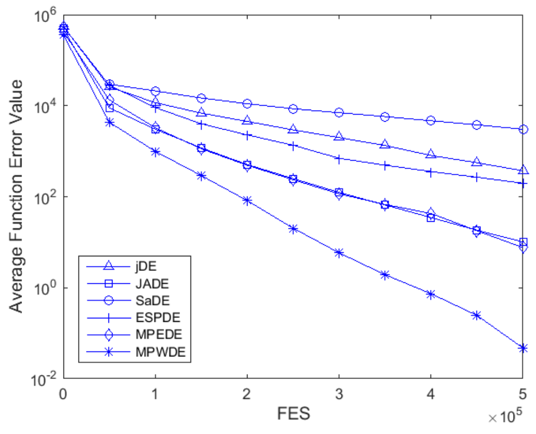

Fig. 6. 30 dimensional f_{11} function

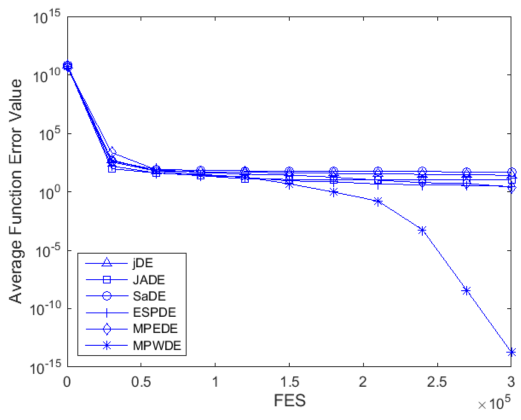

Fig. 7. 30 dimensional f14 function

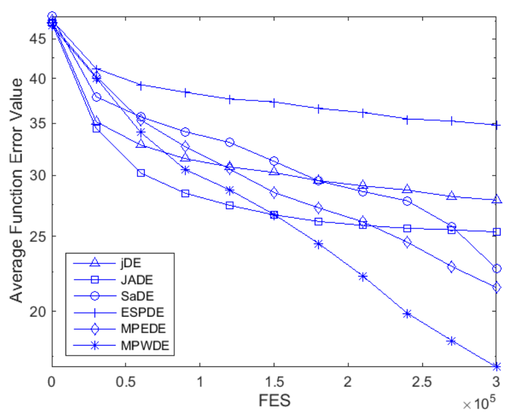

Fig. 8. 50 dimensional f4 function

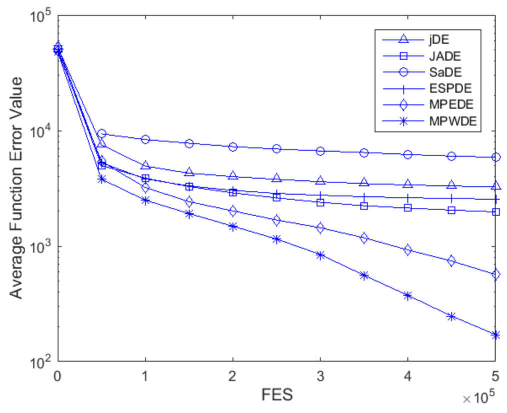

Fig. 9. 50 dimensional f_5 function

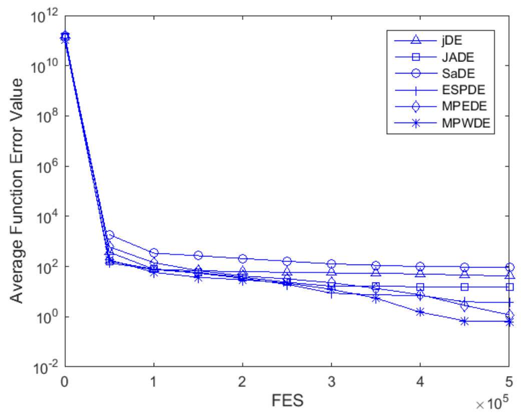

Fig. 10. 50 dimensional f_6 function

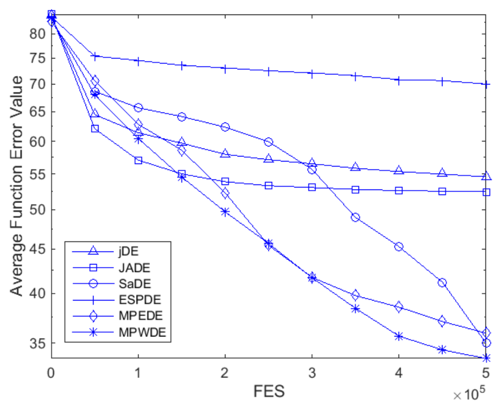

Fig. 11. 50 dimensional f_{11} function

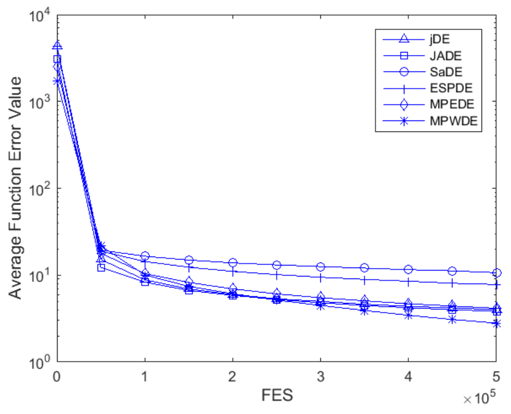

Fig. 12. 50 dimensional f_{13} function

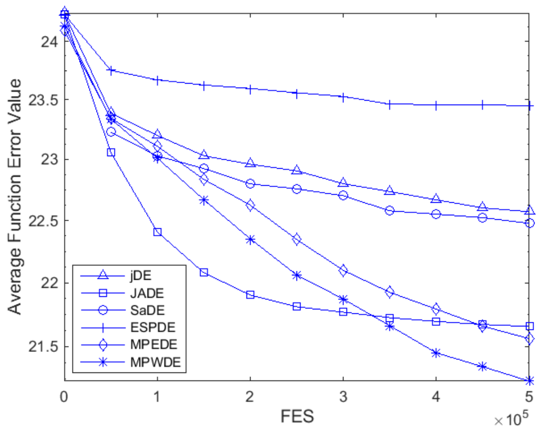

Fig. 13. 50 dimensional f_{14} function

In general, the runtime of the evolution algorithm includes the operation time of the algorithm and the time of evaluating the appropriate function. The jDE, JADE, SaDE, EPSDE, MPEDE and MPWDE algorithms are run 25 times independently on the 14 benchmark functions, recording the average CPU time consumed. AR denotes the acceleration rate and the last row of the table represents the Average AR (*AAR*). Comparison of the average runtime for the six algorithms is shown in Table 5.

Table 5. Comparison of the average runtime(s) of JADE, jDE, SaDE, EPSDE, MPEDE and MPWDE for each test function

Function	JADE	jDE	SaDE	EPSDE	MPEDE	MPWDE
f_1	2.24	1.81	23.89	73.17	2.03	3.12
f_2	2.59	2.03	21.86	75.91	2.66	2.57
f_3	2.96	2.08	23.16	72.01	2.81	2.66
f_4	2.88	2.06	19.49	72.34	2.75	2.57
f_5	3.14	2.32	21.61	72.45	3.14	2.95
f_6	2.52	1.86	20.09	78.05	2.42	2.35
f_7	3.31	2.47	20.78	72.79	2.62	2.51
f_8	3.45	2.48	20.04	69.81	3.06	2.84
f_9	2.64	1.97	20.75	72.72	2.55	2.53
f_{10}	3.34	2.31	19.46	71.13	2.81	2.78
f_{11}	34.23	28.29	45.73	96.67	32.01	28.15
f_{12}	12.38	10.19	27.56	79.62	11.32	10.16
f_{13}	2.99	2.20	18.77	70.35	2.77	2.79
f_{14}	3.82	2.91	20.05	69.79	3.49	3.31
AAR	0.87	1.10	0.22	0.06	0.93	

AAR < 1 and *AAR* > 1 mean that MPWDE is faster and slower than another corresponding algorithm, respectively. As can be seen from Table 5: *AAR* from 0.06 to 1.10, MPWDE is faster than the four algorithms (JADE, SaDE, EPSDE, MPEDE) from the average *AR*. In contrast, MPWDE is slower than jDE. The reason for this phenomenon is that the operator uses dynamic parabola increments and operates directly on the original *CR* variation without having to feed back to the next generation of population. Thus the entire algorithm reduces the running time.

5 Conclusion

It is often unknown, complex, and high dimensional about Practical optimization problem. Because Standard differential evolution algorithm is relatively simple and has less controlled parameters, single strategies which cannot continue to improve its adaptability to the external environment in order to gradually approach the optimal solution to the problem. According to the theory of population diversity, MPWDE is proposed in this paper. Based on the weighted factor of double mutation strategies, the improved algorithm is step by step from global search to local search. The adjustment strategy of the dynamic increment of the *CR* has three merits: (1) the complex parameter feedback process is avoided, (2) the convergence speed of the algorithm is improved, and (3) the contradiction between the convergence rate and the robustness of the search algorithm is improved. Compared with other state-of-the-art algorithms, both the global search ability and the accuracy of the solution are improved in the proposed algorithm.

Acknowledgements

This work is supported by National Natural Science Foundation of China No. 61231002 and No.51075068, by the Foundation of the Fujian Education Department under Grant No. JA15035, by Quanzhou City Science & Technology Program of China under Grant No. 2014Z103, No. 2015Z114 and No. 2018C106R, by the 2016 Postgraduate Innovation Ability Cultivating Projects No. 1611422002.

References

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

[2] B. Liu, L. Wang, Y.-H. Jin, Advances in differential evolution, Control and Decision 22(7)(2007) 721-729.

[3] J. Breast, S. Greiner, B. Boskovoic, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problem, IEEE Transactions on Evolutionary Computation 10(6)(2006) 646-657.

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

[5] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Transactions on Evolutionary Computation 15(1)(2011) 55-66.

[6] J. Zhang, A.C.S., JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13(5)(2009) 945-967.

[7] R.-A. Sarker, S.-M. Elsayed, T. Ray, Differential evolution with dynamic parameters selection for optimization problems, IEEE Transactions on Evolutionary Computation 18(5)(2014) 689-707.

[8] Y. Wang, H.-X. Li, T. Huang, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Applied Soft Computing 18(1)(2014) 232-247.

[9] R. Mallipeddi, P.-N. Suganthan, Q.-K Pan, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing 11(2)(2011) 1679-1696.

[10] K.-M. Sallam, S.-M. Elsayed, R.-A. Sarker, Landscape-based adaptive operator selection mechanism for differential evolution, Information Sciences 418(2017) 383-404.

[11] X. Zhang, X. Zhang, Improving differential evolution by differential vector archive and hybrid repair method for global optimization, Soft Computing 21(23)(2017) 7107-7116.

[12] S.-Y. Park, J.-J. Lee, Stochastic opposition-based learning using a beta distribution in differential evolution, IEEE Transactions on Cybernetics 46(10)(2016) 2184-2194.

[13] S.-Z. Zhao, P.-N. Suganthan, Empirical investigations into the exponential crossover of differential evolutions, Swarm & Evolutionary Computation 9(2013) 27-36.

[14] Y.-J. Zheng, X.-L. Xu, H.-F. Ling, S.-Y. Chen, A hybrid fireworks optimization method with differential evolution operators, Neurocomputing 148(148)(2015) 75-82.

[15] W. Zhu, Y. Tang, J.-A. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Information Sciences 223(2)(2013) 164-191.

[16] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Transactions on Cybernetics 43(6)(2013) 2066-2081.

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

[18] W. Gao, G.-G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Transactions on Cybernetics 44(8)(2014) 1314-1327.

[19] G.-H. Wu, R. Mallipeddi, P.-N. Suganthan, Differential evolution with multi-population based ensemble of mutation strategies, Information Sciences 329(C)(2016) 329-345.

[20] J.-J. Liang, T.-P. Runarsson, E. Mezura-Montes, M. Clerc, P.-N. Suganthan, C.A.C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, Nanyang Technological University, Singapore, Technical Report.