Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 1395025, 12 pages
https://doi.org/10.1155/2017/1395025

Research Article

Memetic Differential Evolution with an Improved Contraction Criterion

Lei Peng,[1,2] Yanyun Zhang,[1] Guangming Dai,[1,2] and Maocai Wang[1,2]

[1] *School of Computer Science, China University of Geosciences, No. 388 Lumo Road, Hongshan District, Wuhan, China*
[2] *Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China*

Correspondence should be addressed to Lei Peng; lei.peng@cug.edu.cn

Received 1 November 2016; Revised 4 March 2017; Accepted 13 March 2017; Published 4 April 2017

Academic Editor: Manuel Graña

Memetic algorithms with an appropriate trade-off between the exploration and exploitation can obtain very good results in continuous optimization. In this paper, we present an improved memetic differential evolution algorithm for solving global optimization problems. The proposed approach, called memetic DE (MDE), hybridizes differential evolution (DE) with a local search (LS) operator and periodic reinitialization to balance the exploration and exploitation. A new contraction criterion, which is based on the improved maximum distance in objective space, is proposed to decide when the local search starts. The proposed algorithm is compared with six well-known evolutionary algorithms on twenty-one benchmark functions, and the experimental results are analyzed with two kinds of nonparametric statistical tests. Moreover, sensitivity analyses for parameters in MDE are also made. Experimental results have demonstrated the competitive performance of the proposed method with respect to the six compared algorithms.

1. Introduction

In 1989, the name of "memetic algorithms" (MAs) [1] was introduced for the first time. In the last two decades, MAs gradually became one of the recent growing areas of research in evolutionary computation. They combine various evolutionary algorithms (EAs) with different LS methods to balance exploration and exploitation. Existing examples of memetic algorithms are NM-BRO [2], MA-LSCh-CMA [3], LBBO [4], IMMA [5], and MPSO [6]. In the framework of MAs, LS operators are used to execute further exploitation for the individuals generated by common EA operations, which is helpful to enhance the EA's capacity of solving complicated problems.

Differential evolution was first proposed by Storn and Price [7] in 1995 to solve global numerical optimization problems over continuous search spaces. It shares some similarities with other EAs. For example, DE works with a population of solutions, called vectors; it uses recombination and mutation operators to generate new vectors and, finally, it has a replacement process to discard the less fit vectors. DE represents solutions with real coding. Some of the differences with respect to other EAs are as follows: DE uses a special mutation operator based on the linear combination of three individuals and uses a uniform crossover operator. It has several attractive features. DE is relatively simple to implement and was demonstrated to be very effective on a large number of cases. In the past few decades, DE has been successfully used in many real-world applications, such as space trajectory design [8–10], hydrothermal optimization [11], underwater glider path planning [12], and vehicle routing problem [13].

Despite its successful applications to different classes of problems in different fields, DE was demonstrated to converge to a fixed point, a level set [10], or a hyperplane not containing the global optimum [14]. Furthermore, in some cases it was shown to have slow local convergence.

In order to overcome these shortcomings, some authors have proposed a hybridization of DE with some local search heuristics. dos Santos Coelho and Mariani [15] proposed a version of memetic DE which combines DE with the generator of chaos sequences and sequential quadratic programming technique (DEC-SQP). In this memetic algorithm, DE with chaos sequences is the global optimizer and SQP

is applied to the best individual to find the local minimum. Noman and Iba [16] proposed an adaptive hill-climbing crossover-based local search operation for enhancing the performance of standard differential evolution (DEahcSPX). Muelas et al. [17] developed MDE-DC which combines DE with multiple trajectory search algorithm (MTS). Neri and Tirronen [18] proposed the scale factor local search differential evolution (SFLSDE). In SFLSDE, golden section search and unidimensional hill-climb local search are applied to detect an optimal value of the scale factor and generate a higher quality offspring. Wang et al. [19] proposed an adaptive MA framework called DE-LS. In DE-LS, self-adaptive differential evolution (SaDE) [20] is the global search method, while covariance matrix adaptation evolution strategy (CMA-ES) [21] and self-adaptive mixed distribution based univariate EDA (MUEDA) [22] are employed as the local search methods. Vasile et al. [10] proposed an inflationary differential evolution algorithm (IDEA), which hybridizes DE with the restarting procedure of Monotonic Basin Hopping (MBH), to solve space trajectory optimization problems. Minisci and Vasile [9] and Di Carlo et al. [8] proposed an adaptive version of inflationary differential evolution algorithm (AIDEA) and a multipopulation version of AIDEA (MP-AIDEA) which automatically adapt the values of four control parameters. Locatelli et al. [23] proposed a memetic differential evolution for disk-packing and sphere-packing problems. In this algorithm, two kinds of local searches (MINOS and SNOPT) are used to detect local minima. Asafuddoula et al. [24] proposed an adaptive hybrid DE algorithm (AH-DEa) which has three features. The first is its use of adaptive crossover rates from a given set of discrete values. The second is an adaptive crossover strategy at different stages of the evolution. The last is the inclusion of a local search strategy to further improve the best solution. Qin et al. [25] proposed an advanced SaDE, which incorporates two different local search chains (Lamarckian and Baldwinian) into SaDE to enhance exploitation capability. Trivedi et al. [26] hybridized DE and GA to solve the unit commitment scheduling problems, in which GA was used to handle the binary unit commitment variables while DE was employed to optimize the continuous power dispatch related variables. In the same year, Li et al. [27] proposed a new hybridization, named DEEP, based on DE framework and the key features of CMA-ES, which generates a trial vector by first using a DE/rand/1/bin strategy followed by an Evolution Path (EP) mutation of CMA-ES.

The focus of this paper is to optimally combine DE global search operators with Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm to improve local search in continuous optimization. A new contraction criterion, which is based on the maximum distances in objective space and decision space, is proposed. When the contraction criterion is satisfied, BFGS starts from the best solution at the current generation. Furthermore, a restart mechanism is employed. If the best solution is not improved during the course of the local search, the population is reinitialized to increase the chance to find the global optimum.

The paper is organized as follows: DE is briefly introduced in Section 2. The proposed DE algorithm with local search and reinitialization is presented in Section 3. The design of the experiments, the results, and the corresponding discussions are included in Section 4. The last section, Section 5, is devoted to conclusions and the future work.

2. A Short Introduction to Differential Evolution

DE is a population-based stochastic parallel optimization method. Each vector (or individual) of the population at t generation is called the target vector, and it will generate one offspring called the trial vector. For example, the ith vector of the population x_i will generate one trial vector u_i. Trial vectors are generated by adding weighted difference vectors to the target vector. This process is referred to as the mutation operator where the target vector is mutated. A crossover step is then applied to produce an offspring which is only accepted if it improves on the fitness of the parent individual. Many variants of standard DE have been proposed, which use different learning strategies and/or recombination operations in the reproduction stage. A general DE variant may be recorded as DE/a/b/c, where "a" denotes the mutation strategy, "b" specifies the number of difference vectors used, and "c" specifies the crossover scheme which may be binomial or exponential. The DE/rand/1/exp is described in Algorithm 1.

3. Proposed Algorithm

In this section, we describe four major operations of the proposed MDE algorithm in detail, including contraction criterion, BFGS search, reinitialization scheme, and boundary constraint handling. The detailed description of MDE is given in Algorithm 2.

3.1. Contraction Criterion. In order to design an effective and efficient hybrid algorithm for global optimization, we need to take advantage of both the exploration capabilities of EA and the exploitation capabilities of LS and combine them in a well-balanced manner. To incorporate BFGS into DE successfully, a triggering condition, called contraction criterion, is needed to decide when the local search has to start. There are several kinds of methods to define a contraction criterion. Qin and Suganthan [20] applies local search method after a fixed number of generations (every 200 generations). Sun et al. [5] starts the LS if the promising solution is not updated in t-consecutive generations. Simon et al. [4] use the minimum fitness in the objective space as the contraction criterion; Di Carlo et al. [8–10] perform LS when the maximum distance in decision space is below a given threshold.

In MDE, we propose a new contraction criterion which combines two criteria: (a) ρ_1 is the improved maximum distance in objective space and (b) ρ_2 is the maximum distance in decision space. The idea of ρ_1 is derived from [28]

$$\rho_1 = \left[\sum_{i=1}^{M} \frac{\left(f_i(x) - f_{\text{avg}}(x) \right)^2}{(M-1)} \right]^{1/2}, \tag{1}$$

where ρ_1 is a measure of the diversity of the population in objective space.

(1) Generate the initial population P, define x_i as the ith individual in P, M is the population size, NFEs is the number of function evaluations in each run, Max_GEN is the maximum generation, Max_NFEs is the number of max function evaluation, D is the number of decision variable, F is the mutation factor, CR is crossover rate.
(2) NFEs = 0
(3) Evaluate the fitness $f(x_i)$ for the each individual in P.
(4) NFEs = NFEs + M
(5) **while** $t \leq$ Max_GEN **and** NFES $\leq$ Max_NFEs **do**
(6) **for** $i = 1$ to M **do**
(7) Select x_{r1}, x_{r2}, x_{r3} and $r1 \neq r2 \neq r3 \neq i$
(8) j = randint$(1, D)$
(9) $L = 0$
(10) $u_i = x_i$
(11) **repeat**
(12) $u_{i,j} = x_{r1,j} + F * (x_{r2,j} - x_{r3,j})$
(13) $j = (j + 1)\%D$
(14) $L = L + 1$
(15) **until** ((randreal$[0, 1) >$ CR) **or** $(L > D)$)
(16) Evaluate the trial vector u_i
(17) NFEs = NFEs + 1
(18) **if** u_i is better than x_i **then**
(19) $x_i = u_i$
(20) **end if**
(21) **end for**
(22) $t = t + 1$
(23) **end while**

Algorithm 1: DE with rand/1/exp.

The distance in decision space is defined as

$$\rho_2 = \max\left(\left\|x_i - x_j\right\|\right), \quad \forall x_i, x_j \in P, \tag{2}$$

where $\|\cdot\|$ is the Euclidean distance. ρ_2 is a measure of the diversity of the population in decision space.

3.2. BFGS Search. In MDE, the local search utilizes the better solutions obtained by the global search to update the population of MDE and thus enhances MDE's exploitation ability to find the best solution. In MDE, we use the BFGS algorithm as the local search method. BFGS is one of the quasi-Newton methods which do not need the precise Hessian matrix and is able to approximate it based on the individual successive gradients. BFGS is considered as the most effective and popular quasi-Newton method and has been proven to have good performance even for nonsmooth optimizations. The details can be found in [29].

3.3. Reinitialization Scheme. If the best solution has not been improved after local search, a reinitialization of the whole population is used to give the algorithms more opportunities to find the global optimum. Simon et al. [4] proposed a partial reinitialization of the population. Every 20 generations, the algorithm selects the best M individuals from a temporary population of $2M + 2$ individuals as the reinitialization pool. Sun et al. [5] chose the individuals, which have the largest distances from the local optima, from a temporary population to form the next population. Zamuda et al. [30] proposed a population size reduction method as the reinitialization scheme. In MDE, we apply a simple reinitialization scheme described in Algorithm 3. If the result of the local search does not improve the best individual in the population, a reinitialization of the population is triggered. A counter C keeps track of the number of restarts. For $C < C_{\max}$, where $C_{\max}$ is user-defined, M individuals are generated randomly in the search space, drawing samples from a uniform distribution. For $C \geq C_{\max}$, $2M/3$ individuals in the population are initialized randomly in the search space, while the rest are initialized by a normal distribution which takes the best individual as the center and $(U_i - L_i)/50$ as the standard deviation. Algorithm 3 summarises the reinitialization procedure. The function *randreal* draws samples from a uniform distribution while function *Gaussian* draws samples from a normal distribution and $[L_i, U_i]$ are the lower and upper boundaries on x_i.

3.4. Boundary Constraint Handling. After mutation and crossover, each generated trial vector u_i undergoes boundary constraint check. If some variables of u_i are out of the boundary, a repair method is applied as follows:

$$x_i = \text{randreal}\left(L_i, U_i\right), \quad \text{if } x_i < L_i \text{ or } x_i > U_i, \tag{3}$$

where randreal (L_i, U_i) can generate a random real number from $[L_i, U_i]$.

4. Experimental Results

In order to verify the performance of MDE, we select the 21 nonnoisy benchmark functions from CEC2005 special

(1) Generate the initial population P, define x_i as the ith vector in P,
x_{best} is the best vector in the current generation,
x_{min} is the global best vector in the generation. M is the population size,
NFEs is the number of function evaluations in each run, Max_GEN is the maximum generation,
Max_NFEs is the number of max function evaluation,
D is the number of decision variable, F is the mutation factor, CR is crossover rate,
ρ_1 and ρ_2 are the contraction criterion, Flag is the restart mark.
(2) NFEs = 0
(3) Evaluate the fitness $f(x_i)$ for the each individual in P.
NFEs = NFEs + M.
(4) Flag = 0, count = 0
(5) **while** $t \leq$ Max_GEN **and** NFES $\leq$ Max_NFEs **do**
(6) **Global search using DE**
(7) **for** $i = 1$ to M **do**
(8) CR $\in N(0.8, 0.1)$, $F \in N(0.5, 0.1)$
(9) Select x_{r1}, x_{r2}, x_{r3} and $r1 \neq r2 \neq r3 \neq i$
(10) $u_i = x_i$
(11) **if** NFES $< 0.2 \times$ Max_NFEs **then**
(12) Using DE/rand/1/bin to generate u_i
(13) **else**
(14) Using DE/rand/1/exp to generate u_i
(15) **end if**
(16) Evaluate the trial vector u_i. NFEs = NFEs + 1
(17) **if** u_i is better than x_i **then**
(18) $x_i = u_i$
(19) **end if**
(20) **end for**
(21) **if** $f(x_{\text{best}}) < f(x_{\text{min}})$ **then**
(22) Replace x_{min} with x_{best}.
(23) **end if**
(24) **Local search using BFGS**
(25) Calculate the contraction criterion as described in Section 3.1
(26) **if** ($\rho_1 < \rho_{1,\text{max}}$ **or** $\rho_2 < \rho_{2,\text{max}}$) **then**
(27) Pick up the x_{best} as the initial point of the local search.
(28) Apply BFGS (x_{best}) to find the resultant new local optimum x_{local} as described in Section 3.2.
(29) **if** $f(x_{\text{local}}) < f(x_{\text{best}})$ **then**
(30) Replace x_{best} with x_{local}.
(31) NFEs = NFEs + 1
(32) Flag = 0
(33) **else**
(34) Flag = 1
(35) **end if**
(36) **if** $f(x_{\text{best}}) < f(x_{\text{min}})$ **then**
(37) Replace x_{min} with x_{best}.
(38) **end if**
(39) **Restart mechanism**
(40) **if** Flag == 1 **then**
(41) Run the re-initialization to create a new population P' as described in Section 3.3.
(42) Reinitialize x_{best}.
(43) **end if**
(44) $t = t + 1$
(45) **end if**
(46) **end while**

Algorithm 2: Pseudocode of MDE.

session on real-parameter optimization (excluding noisy functions $F4$, $F17$, $F24$, and $F25$) since MDE has no ability to handle functions with noisy landscapes. The details about these functions can be found in [31]. We compare MDE with six peer algorithms, including CLPSO [32], GL-25 [33], CMA-ES [21], LBBO [4], SFLSDE [18], and L-SHADE [34].

```
(1) std = (U_i − L_i)/50.0
(2) if C < C_max then
(3)     Generate P randomly.
(4) else
(5)     for i = 1 to 2/3 ∗ M do
(6)         x_i = randreal (L_i, U_i)
(7)     end for
(8)     for i = 2/3 ∗ M to M do
(9)         x_i = Gaussian (x_best, std)
(10)    end for
(11) end if
```

Algorithm 3: Pseudocode of reinitialization scheme.

4.1. Experimental Setup. For each algorithm on each benchmark problem, we conduct 25 independent runs and limit each run to $10000 * D$ max function evaluations, where D is the problem dimension ($D = 10$, $D = 30$, and $D = 50$). The performance of the algorithms is evaluated in terms of *function error value* [31], defined as $f(x) - f(x^*)$, where x^* is the global optimum of the test function. The mean and the standard deviation of the *function error values* are recorded. The parameters of MDE are set as $M = 30$, $\rho_{1,\max} = 2.0$, $\rho_{2,\max} = 2.0$, $C_{\max} = 3$, $CR \in N(0.8, 0.1)$, and $F \in N(0.5, 0.1)$; the mutation and crossover strategies are the same as those in [24]. For the other six algorithms, we use the same parameter settings in their original papers.

4.2. Performance Criteria. To effectively analyze the results, two nonparametric statistical tests, as similarly done in [35, 36], are used in the experiments. (i) Wilcoxon's signed-rank test at $\alpha = 0.05$ is performed to test the statistical significance of the experimental results between two algorithms on both single-problem and multiproblem. (ii) Friedman's test is employed to obtain the average rankings of all the compared algorithms. Wilcoxon's signed-rank test on single-problem is calculated by Matlab, while Wilcoxon's signed-rank test on multiproblem and Friedman test are calculated by the software of KEEL [37].

4.3. Comparison between the Other Six Algorithms and MDE. Table 1 shows the results of MDE and the other six algorithms on the 10-dimensional benchmarks. It can be seen that MDE performs significantly better than CLPSO, GL-25, CMA-ES, LBBO, SFLSDE, and L-SHADE on 15, 16, 17, 7, 8, and 8 test functions. And CLPSO, GL-25, CMA-ES, LBBO, SFLSDE, and L-SHADE win on 4, 4, 3, 5, 8, and 8 test functions, respectively. MDE obtains similar results with the other six algorithms in 2, 1, 1, 9, 5, and 5 cases. Additionally, the results of the multiple-problem statistical analysis are shown in Table 4. It can be seen that MDE can obtain higher R^+ values than R^- values in all cases, where R^+ is the sum of ranks for the functions on which MDE outperformed the compared algorithm, and R^- is the sum of ranks for the opposite [36]. According to Wilcoxon's test at $\alpha = 0.05$ and $\alpha = 0.1$, there are significant differences in three cases (MDE versus CLPSO, MDE versus GL-25, and MDE versus CMA-ES),

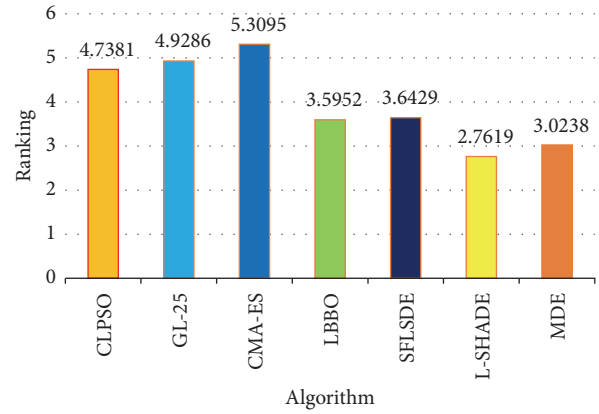

Figure 1: Average rankings of the seven algorithms by Friedman test for all functions at $D = 10$.

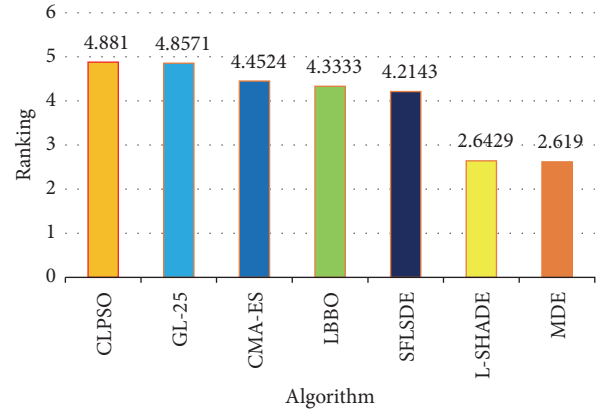

Figure 2: Average rankings of the seven algorithms by Friedman test for all functions at $D = 30$.

which means that in those cases MDE is significantly better than CLPSO, GL-25, and CMA-ES. In addition, Friedman's test is employed to evaluate the significant differences of all the compared algorithms. As shown in Figure 1, MDE gets the second average ranking value, while L-SHADE gets the first average ranking values on the 10-dimensional problems.

Table 2 shows that MDE performs significantly better than the other six compared algorithms in the majority of the test functions. For example, MDE wins in 12 cases, only loses in 3 cases, and ties in 6 cases, compared with SFLSDE. We can also find that MDE obtains much better solutions than LBBO and CLPSO. As for L-SHADE, MDE wins in 8 cases and loses in 10 cases. According to the results of the multiple-problem statistical analysis shown in Table 5, it can be seen that MDE can obtain higher R^+ values than R^- values in all cases. According to Wilcoxon's test at $\alpha = 0.05$ and $\alpha = 0.1$, there are significant differences in four cases (MDE versus CLPSO, MDE versus GL-25, MDE versus CMA-ES, and MDE versus SFLSDE), which means that in those cases MDE is significantly better than CLPSO, GL-25, CMA-ES, and SFLSDE. And from Table 5, we can find that L-SHADE and MDE have comparable results. Moreover, Figure 2 shows that MDE performs the first average ranking value and

Table 1: Comparison of Mean Error and standard deviation between MDE and other six EAs over 25 independent runs on twenty-one 10-dimensional test functions.

Prob	CLPSO	GL-25	CMA-ES	LBBO	SFLSDE	L-SHADE	MDE
F01*	3.32E + 00 ±	1.71E − 26 ±	3.17E − 27 ±	0.00E + 00 ±	6.50E − 09 ±	6.66E − 08 ±	0.00E + 00 ±
	1.96E + 00−	4.40E − 26−	1.82E − 27−	0.00E + 00=	1.37E − 08−	1.05E − 07−	0.00E + 00
F02	7.42E − 02 ±	2.73E − 28 ±	6.17E − 27 ±	0.00E + 00 ±	0.00E + 00 ±	0.00E + 00 ±	0.00E + 00 ±
	5.95E − 02−	4.79E − 28−	2.18E − 27−	0.00E + 00=	0.00E + 00=	0.00E + 00=	0.00E + 00
F03	3.54E + 05 ±	2.49E + 04 ±	3.94E − 23 ±	0.00E + 00 ±	9.54E + 03 ±	0.00E + 00 ±	0.00E + 00 ±
	1.45E + 05−	1.12E + 04−	2.17E − 23−	0.00E + 00=	8.05E + 03−	0.00E + 00=	0.00E + 00
F05	6.29E + 00 ±	5.21E − 05 ±	4.82E − 11 ±	3.06E − 03 ±	0.00E + 00 ±	0.00E + 00 ±	1.13E − 02 ±
	5.49E + 00−	2.28E − 04+	6.54E − 12+	3.68E − 03+	0.00E + 00+	0.00E + 00+	8.32E − 03
F06	9.25E − 01 ±	2.30E + 00 ±	9.57E − 01 ±	1.53E − 06 ±	1.59E − 01 ±	0.00E + 00 ±	0.00E + 00 ±
	7.47E − 01−	6.43E − 01−	1.74E + 00−	4.89E − 06=	7.97E − 01=	0.00E + 00=	0.00E + 00
F07	2.82E − 01 ±	1.07E − 01 ±	1.21E − 02 ±	1.06E − 01 ±	1.27E + 03 ±	3.84E − 03 ±	1.87E − 02 ±
	9.68E − 02−	2.80E − 02−	1.16E − 02+	2.15E − 01=	1.70E − 01−	6.20E − 03+	1.75E − 02
F08	2.04E + 01 ±	2.04E + 01 ±	2.00E + 01 ±	2.00E + 01 ±	2.04E + 01 ±	2.01E + 01 ±	2.00E + 01 ±
	8.71E − 02−	8.87E − 02−	7.49E − 02=	2.95E − 09=	9.14E − 02−	1.03E − 01−	0.00E + 00
F09*	5.47E + 00 ±	6.83E + 00 ±	1.26E + 02 ±	9.80E − 12 ±	2.91E + 00 ±	8.37E − 01 ±	4.54E + 00 ±
	1.61E + 00−	4.58E + 00−	4.21E + 01−	1.31E − 11+	2.27E + 00+	7.90E − 01+	1.99E + 00
F10	8.53E + 00 ±	1.36E + 01 ±	7.51E + 01 ±	1.32E + 01 ±	7.41E + 00 ±	2.47E + 00 ±	4.10E + 00 ±
	1.70E + 00−	9.53E + 00−	1.01E + 02−	3.76E + 00−	2.95E + 00−	1.04E + 00+	1.16E + 00
F11	5.01E + 00 ±	3.20E + 00 ±	2.42E + 00 ±	5.04E + 00 ±	1.77E + 00 ±	4.12E + 00 ±	6.69E + 00 ±
	7.30E − 01+	9.45E − 01+	1.30E + 00+	1.17E + 00+	1.36E + 00+	7.50E − 01+	1.56E + 00
F12	1.38E + 02 ±	1.04E + 01 ±	6.92E + 03 ±	2.59E − 02 ±	1.15E + 02 ±	2.40E + 00 ±	0.00E + 00 ±
	9.66E + 01−	8.81E − 01−	1.21E + 04−	1.20E − 01−	3.33E + 02−	4.36E + 00−	0.00E + 00
F13	4.56E − 01 ±	9.68E − 01 ±	1.02E + 00 ±	4.16E − 01 ±	3.66E − 01 ±	2.31E − 01 ±	4.05E − 01 ±
	7.95E − 02−	4.21E − 01−	4.47E − 01−	1.52E − 01=	1.02E − 01+	3.67E − 02+	1.26E − 01
F14	3.15E + 00 ±	2.86E + 00 ±	4.87E + 00 ±	3.33E + 00 ±	2.86E + 00 ±	2.40E + 00 ±	3.77E + 00 ±
	2.24E − 01+	4.76E − 01+	1.87E − 01−	3.55E − 01+	3.29E − 01+	2.93E − 01+	3.30E − 01
F15	9.30E + 00 ±	3.79E + 02 ±	5.30E + 02 ±	1.70E − 12 ±	2.72E + 02 ±	1.78E + 02 ±	6.04E + 01 ±
	2.09E + 01+	6.21E + 01−	2.76E + 02−	1.89E − 12+	1.53E + 02−	1.90E + 02−	7.57E + 01
F16	1.15E + 02 ±	9.69E + 01 ±	2.03E + 02 ±	1.27E + 02 ±	1.11E + 02 ±	9.26E + 01 ±	1.15E + 02 ±
	1.04E + 01=	1.10E + 01+	2.23E + 02−	1.51E + 01−	1.10E + 01+	2.66E + 00+	1.17E + 01
F18	6.57E + 02 ±	7.88E + 02 ±	8.26E + 02 ±	7.60E + 02 ±	5.20E + 02 ±	6.00E + 02 ±	5.51E + 02 ±
	1.18E + 02−	5.80E + 01−	3.85E + 02−	1.61E + 02−	2.55E + 02+	2.50E + 02=	2.36E + 02
F19	6.11E + 02 ±	8.00E + 02 ±	7.73E + 02 ±	7.71E + 02 ±	5.33E + 02 ±	6.40E + 02 ±	4.95E + 02 ±
	1.39E + 02−	1.86E − 02−	3.44E + 02−	1.49E + 02−	2.21E + 02=	2.38E + 02−	2.38E + 02
F20	6.64E + 02 ±	7.80E + 02 ±	7.06E + 02 ±	7.11E + 02 ±	4.54E + 02 ±	6.20E + 02 ±	5.17E + 02 ±
	1.52E + 02−	1.00E + 02−	2.86E + 02−	1.97E + 02−	2.15E + 02+	2.45E + 02−	2.37E + 02
F21	4.49E + 02 ±	8.00E + 02 ±	8.54E + 02 ±	5.35E + 02 ±	5.64E + 02 ±	4.88E + 02 ±	4.45E + 02 ±
	1.11E + 02=	8.37E − 14−	2.64E + 02−	2.66E + 02−	1.89E + 02−	1.90E + 02−	1.83E + 02
F22	7.47E + 02 ±	6.72E + 02 ±	7.69E + 02 ±	6.51E + 02 ±	7.63E + 02 ±	7.45E + 02 ±	6.90E + 02 ±
	9.64E + 01−	1.90E + 02=	2.53E + 01−	2.24E + 02=	2.85E + 01=	1.20E + 01=	1.64E + 02
F23	5.58E + 02 ±	9.74E + 02 ±	9.89E + 02 ±	6.40E + 02 ±	6.62E + 02 ±	6.44E + 02 ±	6.42E + 02 ±
	6.83E + 01+	1.63E + 01−	2.59E + 02−	9.69E + 01=	1.87E + 02=	1.24E + 02−	1.46E + 02
−	15	16	17	7	8	8	/
+	4	4	3	5	8	8	/
=	2	1	1	9	5	5	/

∗ indicates that when six algorithms obtain the global optimum, the intermediate results are reported at NFEs = 10000. "−," "+," and "=" denote that the performance of this algorithm is, respectively, worse than, better than, and similar to MDE according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

Table 2: Comparison of Mean Error and standard deviation between MDE and other six EAs over 25 independent runs on twenty-one 30-dimensional test functions.

Prob	CLPSO	GL-25	CMA-ES	LBBO	SFLSDE	L-SHADE	MDE
F01*	2.81E + 01 ±	7.05E − 24 ±	2.05E − 25 ±	0.00E + 00 ±	1.36E − 11 ±	3.52E − 05 ±	1.36E − 14 ±
	6.52E + 00−	3.18E − 23=	6.01E − 26=	0.00E + 00+	1.43E − 11−	2.87E − 05−	2.48E − 14
F02	8.66E + 02 ±	5.46E + 01 ±	6.36E − 25 ±	2.11E − 08 ±	3.78E − 09 ±	6.82E − 15 ±	2.07E − 13 ±
	1.75E + 02−	8.48E + 01−	2.18E − 25+	8.29E − 08−	4.84E − 09−	1.89E − 14+	8.51E − 14
F03	1.62E + 07 ±	2.13E + 06 ±	5.38E − 21 ±	3.97E + 02 ±	3.97E + 05 ±	2.86E − 13 ±	0.00E + 00 ±
	4.99E + 06−	8.45E + 05−	1.66E − 21−	6.54E + 02−	2.40E + 05−	5.04E − 13−	0.00E + 00
F05	3.97E + 03 ±	2.48E + 03 ±	3.34E − 10 ±	2.69E + 03 ±	1.07E + 03 ±	1.32E − 11 ±	3.05E + 02 ±
	4.11E + 02−	2.08E + 02−	7.85E − 11+	7.88E + 02−	6.35E + 02−	7.71E − 12+	9.81E + 01
F06	6.09E + 00 ±	2.17E + 01 ±	4.78E − 01 ±	2.15E − 01 ±	4.78E − 01 ±	1.00E − 13 ±	0.00E + 00 ±
	5.44E + 00−	1.53E + 00−	1.32E + 00=	1.03E + 00−	1.32E + 00−	5.51E − 14−	0.00E + 00
F07	4.85E − 01 ±	1.37E − 02 ±	1.58E − 03 ±	9.77E − 02 ±	4.70E + 03 ±	0.00E + 00 ±	6.71E − 14 ±
	8.90E − 02−	1.10E − 02−	4.92E − 03−	2.66E − 01−	1.73E + 00−	0.00E + 00+	2.83E − 14
F08	2.10E + 01 ±	2.10E + 01 ±	2.03E + 01 ±	2.00E + 01 ±	2.10E + 01 ±	2.03E + 01 ±	2.00E + 01 ±
	5.95E − 02−	5.19E − 02−	5.70E − 01−	1.25E − 04−	4.53E − 02−	3.47E − 01−	0.00E + 00
F09*	3.21E + 01 ±	4.84E + 01 ±	4.28E + 02 ±	2.48E − 07 ±	3.37E + 01 ±	4.10E + 01 ±	2.69E + 01 ±
	5.26E + 00−	3.62E + 01−	1.13E + 02−	3.31E − 07+	7.65E + 00−	5.83E + 00−	1.11E + 01
F10	1.05E + 02 ±	1.74E + 02 ±	4.95E + 01 ±	1.73E + 02 ±	4.34E + 01 ±	6.53E + 00 ±	4.07E + 01 ±
	1.35E + 01−	1.22E + 01−	1.29E + 01−	3.10E + 01−	1.27E + 01=	1.60E + 00+	7.91E + 00
F11	2.56E + 01 ±	3.31E + 01 ±	7.43E + 00 ±	2.63E + 01 ±	1.72E + 01 ±	2.64E + 01 ±	2.80E + 01 ±
	1.55E + 00+	7.73E + 00−	2.36E + 00+	2.75E + 00+	3.30E + 00+	1.33E + 00+	3.08E + 00
F12	1.78E + 04 ±	7.13E + 03 ±	1.11E + 04 ±	1.55E + 01 ±	9.72E + 03 ±	8.97E + 02 ±	1.93E + 02 ±
	5.59E + 03−	5.05E + 03−	9.70E + 03−	3.49E + 01=	8.88E + 03−	1.27E + 03−	4.79E + 02
F13	2.12E + 00 ±	5.28E + 00 ±	3.54E + 00 ±	1.94E + 00 ±	2.02E + 00 ±	1.24E + 00 ±	1.55E + 00 ±
	2.68E − 01−	4.08E + 00−	7.71E − 01−	4.23E − 01−	5.91E − 01−	1.06E − 01+	3.51E − 01
F14	1.28E + 01 ±	1.29E + 01 ±	1.46E + 01 ±	1.29E + 01 ±	1.27E + 01 ±	1.18E + 01 ±	1.38E + 01 ±
	1.80E − 01+	4.63E − 01+	2.91E − 01−	2.63E − 01+	5.54E − 01+	3.43E − 01+	3.18E − 01
F15	6.15E + 01 ±	3.04E + 02 ±	4.09E + 02 ±	7.90E + 01 ±	3.04E + 02 ±	3.84E + 02 ±	3.23E + 02 ±
	5.70E + 01+	2.00E + 01=	2.21E + 02=	1.37E + 02+	9.27E + 01=	4.73E + 01−	1.02E + 02
F16	1.71E + 02 ±	1.28E + 02 ±	4.32E + 02 ±	1.74E + 02 ±	1.48E + 02 ±	2.39E + 01 ±	9.89E + 01 ±
	2.88E + 01−	9.13E + 01=	3.57E + 02−	3.52E + 01−	1.19E + 02=	2.60E + 00+	2.61E + 01
F18	8.97E + 02 ±	9.07E + 02 ±	9.28E + 02 ±	9.16E + 02 ±	9.05E + 02 ±	9.03E + 02 ±	8.91E + 02 ±
	7.87E + 01−	1.37E + 00=	1.18E + 02=	3.51E + 01−	1.36E + 00=	1.81E − 01=	4.10E + 01
F19	9.14E + 02 ±	9.06E + 02 ±	9.04E + 02 ±	9.21E + 02 ±	9.06E + 02 ±	9.03E + 02 ±	8.93E + 02 ±
	1.73E + 00−	1.48E + 00=	2.83E − 01=	2.53E + 01−	1.83E + 00=	1.94E − 01=	4.18E + 01
F20	9.14E + 02 ±	9.07E + 02 ±	9.21E + 02 ±	9.24E + 02 ±	9.06E + 02 ±	9.03E + 02 ±	9.02E + 02 ±
	1.23E + 00−	1.49E + 00=	8.59E + 01=	3.14E + 00−	4.11E + 00=	2.07E − 01=	3.18E + 01
F21	5.00E + 02 ±	5.00E + 02 ±	5.12E + 02 ±	5.00E + 02 ±	5.04E + 02 ±	5.00E + 02 ±	5.00E + 02 ±
	5.42E − 13−	4.27E − 13−	6.00E + 01−	1.07E − 08−	1.75E + 01−	2.89E − 13−	2.54E − 13
F22	9.70E + 02 ±	9.27E + 02 ±	8.27E + 02 ±	1.06E + 03 ±	8.71E + 02 ±	8.42E + 02 ±	9.33E + 02 ±
	1.17E + 01−	8.14E + 00+	1.82E + 01+	3.65E + 01−	1.88E + 01+	1.81E + 01+	1.40E + 01
F23	5.34E + 02 ±	5.34E + 02 ±	5.37E + 02 ±	5.90E + 02 ±	7.07E + 02 ±	5.34E + 02 ±	5.34E + 02 ±
	1.15E − 04+	4.63E − 04+	4.41E + 00−	9.52E + 00−	1.57E + 02−	5.77E − 13+	5.38E − 04
−	17	12	11	15	12	8	/
+	4	3	4	5	3	10	/
=	0	6	6	1	6	3	/

∗ indicates that when six algorithms obtain the global optimum, the intermediate results are reported at NFEs = 30000. "−," "+," and "=" denote that the performance of this algorithm is, respectively, worse than, better than, and similar to MDE according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

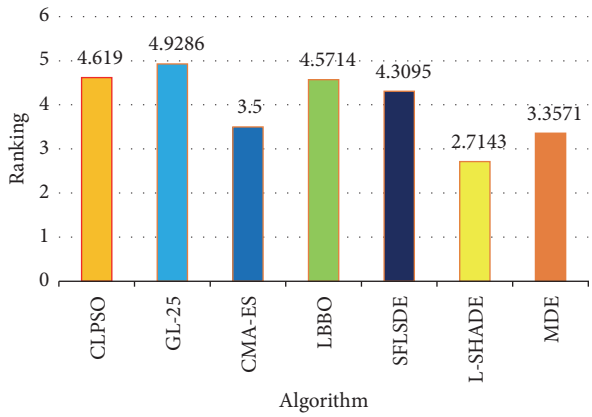

FIGURE 3: Average rankings of the seven algorithms by Friedman test for all functions at $D = 50$.

L-SHADE obtains the second average ranking values on the 30-dimensional problems by Friedman's test.

Table 3 shows that MDE also performs significantly better than five compared algorithms in the majority of the test functions except L-SHADE. For example, MDE wins in 15 cases, only loses in 3 cases, and ties in 3 cases, compared with LBBO. When comparing with L-SHADE, MDE wins in 8 cases and loses in 12 cases. Table 6 shows that MDE can perform higher R^+ values than R^- values in five cases. According to Wilcoxon's test at $\alpha = 0.05$ and $\alpha = 0.1$, there are significant differences in two cases (MDE versus GL-25 and MDE versus LBBO), which means that in those cases MDE is significantly better than GL-25 and LBBO. Figure 3 shows that L-SHADE obtains better average ranking values than the other six algorithms on 50-dimensional problems by Friedman's test.

In general, according to the analysis above, we can conclude that MDE and L-SHADE have better average rankings among the seven algorithms on 21 benchmark problems for all of three different dimensions. The performance of MDE is comparable to L-SHADE on 10- and 30-dimensional problems, while L-SHADE is better than MDE on 50-dimensional problems because of the larger initial population and linear population size reduction.

4.4. Influence of Contraction Criterion. In the previous experiments the recommended initial $\rho_{1,\max} = 2.0$ and $\rho_{2,\max} = 2.0$ are used. In order to test the influence of different initial contraction criterion values to the enhanced performance of MDE, in this section, MDE is tested with different initial $\rho_{1,\max}$ and $\rho_{2,\max}$ values. The initial values are set as $\rho_{1,\max} = \{1.0, 2.0, 3.0\}$ and $\rho_{2,\max} = \{1.0, 2.0, 3.0\}$. All other parameters are not changed as described in Section 4.1. Nine groups of experiments with different combinations between $\rho_{1,\max}$ and $\rho_{2,\max}$ are done. $\rho_{1.0,1.0}$ means the value of parameters $\rho_{1,\max} = 1.0$ and $\rho_{2,\max} = 1.0$. The statistical results by Friedman's test with all initial values are shown in Table 7.

From Table 7, MDE owes the best average ranking value at $\rho_{2.0,2.0}$ than the other 8 groups on both 10-dimensional and 30-dimensional test functions. On 50-dimensional test functions, $\rho_{1.0,1.0}$ is the better choice to MDE. In general, we can conclude that it is better to set smaller contraction criterion values as the dimension of test functions increases.

4.5. Influence of Parameter $C_{\max}$. The experiment is to test the influence of parameter $C_{\max}$ in MDE. Friedman's test results are shown in Table 8, where the values of $C_{\max}$ are set as $C_{\max} = \{3, 5, 7, 9, 11, 13, 15\}$ in Table 8. All other parameters are kept unchanged as described in Section 4.1. In addition, all experiments are conducted for 25 independent runs for each function.

It can be seen from Table 8 that MDE with $C_{\max} = 3.0$ gets the better average ranking value than the other six cases at $D = 10$. At $D = 30$, $C_{\max} = 5.0$ is the best choice and $C_{\max} = 3.0$ is the second best choice to MDE. On 50-dimensional test functions, $C_{\max} = 3.0$ is the better choice parameter to MDE. Generally speaking, the small $C_{\max}$ value such as 3 or 5 is good to enhance the performance of the MDE algorithm.

4.6. Influence of Population Size M. To analyze the influence of the population size M, different values of M are tested in a set of experiments. Friedman's test results are shown in Table 9, where the values of M are set as $\{30, 60, 90, 120, 150\}$. All other parameters are kept unchanged as described in Section 4.1.

From Table 9, MDE with $M = 30$ ranks the first both at $D = 10$ and $D = 30$, while MDE with $M = 90$ performs the best at $D = 50$. From the results, it can be concluded that, as the dimension increases, a properly increased population size M can enhance the search capability of MDE.

5. Conclusion

A memetic differential evolution, MDE, has been introduced in this paper. MDE uses a new contraction criterion to decide when the local search starts. In addition, MDE includes the global and local search operators, along with reinitialization, to improve performance.

To evaluate the performance of MDE, 21 benchmark functions with different characteristics are chosen for test. The results show that (i) MDE can obtain better or at least comparable results, compared with the other six algorithms; (ii) small contraction criterion values and $C_{\max}$ can enhance the performance of MDE in terms of the quality of the final results; (iii) a large population size is good to MDE as the dimension increases.

In this paper, some preliminary experiments have been performed to verify its effect on the performance of MDE. In our future work, MDE will be tested on some real-world applications problems. Moreover, we believe that some other local search algorithms and adaptive population size strategy can also be used in MDE.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Table 3: Comparison of Mean Error and standard deviation between MDE and other six EAs over 25 independent runs on twenty-one 50-dimensional test functions.

Prob	CLPSO	GL-25	CMA-ES	LBBO	SFLSDE	L-SHADE	MDE
F01	$0.00E+00\pm$	$1.48E-23\pm$	$4.46E-25\pm$	$1.46E-10\pm$	$4.77E-14\pm$	$0.00E+00\pm$	$0.00E+00\pm$
	$0.00E+00=$	$5.81E-23-$	$8.18E-26-$	$1.23E-10-$	$2.13E-14-$	$0.00E+00=$	$0.00E+00$
F02	$8.94E+03\pm$	$1.54E+03\pm$	$6.51E-24\pm$	$4.42E-07\pm$	$1.24E+00\pm$	$2.11E-13\pm$	$2.73E-13\pm$
	$1.24E+03-$	$1.10E+03-$	$1.70E-24+$	$2.80E-07-$	$8.43E-01-$	$5.81E-14+$	$9.14E-14$
F03	$4.35E+07\pm$	$5.50E+06\pm$	$4.26E-20\pm$	$1.57E+04\pm$	$1.48E+06\pm$	$1.26E+03\pm$	$0.00E+00\pm$
	$1.05E+07-$	$2.00E+06-$	$8.09E-21-$	$8.05E+03-$	$5.79E+05-$	$1.44E+03-$	$0.00E+00$
F05	$9.45E+03\pm$	$5.70E+03\pm$	$1.40E-01\pm$	$8.53E+03\pm$	$3.33E+03\pm$	$2.09E+02\pm$	$1.19E+03\pm$
	$8.94E+02-$	$5.04E+02-$	$6.98E-01+$	$1.58E+03-$	$7.16E+02-$	$2.02E+02+$	$4.08E+02$
F06	$1.43E+01\pm$	$4.95E+01\pm$	$4.78E-01\pm$	$3.97E+01\pm$	$2.72E+01\pm$	$1.30E-01\pm$	$0.00E+00\pm$
	$1.54E+01-$	$2.14E+01-$	$1.32E+00=$	$7.63E+01-$	$3.19E+01-$	$3.71E-01-$	$0.00E+00$
F07	$3.58E-01\pm$	$6.40E-02\pm$	$1.58E-03\pm$	$4.78E-01\pm$	$6.20E+03\pm$	$2.84E-14\pm$	$2.17E-13\pm$
	$5.69E-02-$	$5.49E-02-$	$4.38E-03-$	$3.03E-01-$	$7.88E-13-$	$0.00E+00+$	$5.31E-14$
F08	$2.11E+01\pm$	$2.11E+01\pm$	$2.05E+01\pm$	$2.00E+01\pm$	$2.11E+01\pm$	$2.04E+01\pm$	$2.00E+01\pm$
	$4.65E-02-$	$4.06E-02-$	$7.13E-01-$	$1.16E-02-$	$3.53E-02-$	$4.43E-01-$	$0.00E+00$
F09	$0.00E+00\pm$	$5.37E+01\pm$	$6.87E+02\pm$	$9.05E-05\pm$	$4.38E-01\pm$	$5.56E-09\pm$	$0.00E+00\pm$
	$0.00E+00=$	$1.23E+01-$	$1.74E+02-$	$2.41E-04=$	$1.12E+00-$	$9.78E-09-$	$0.00E+00$
F10	$2.62E+02\pm$	$2.41E+02\pm$	$9.56E+01\pm$	$3.70E+02\pm$	$9.15E+01\pm$	$1.28E+01\pm$	$9.78E+01\pm$
	$3.16E+01-$	$1.47E+02-$	$1.92E+01=$	$4.89E+01-$	$4.17E+01+$	$2.30E+00+$	$1.51E+01$
F11	$5.09E+01\pm$	$6.49E+01\pm$	$1.15E+01\pm$	$5.45E+01\pm$	$4.22E+01\pm$	$5.23E+01\pm$	$5.90E+01\pm$
	$2.13E+00+$	$1.08E+01-$	$3.69E+00+$	$2.84E+00+$	$1.62E+01+$	$2.12E+00+$	$4.90E+00$
F12	$6.73E+04\pm$	$4.40E+04\pm$	$3.03E+04\pm$	$1.01E+03\pm$	$2.75E+04\pm$	$6.89E+03\pm$	$1.35E+03\pm$
	$1.39E+04-$	$2.00E+04-$	$2.59E+04-$	$1.26E+03=$	$2.29E+04-$	$6.05E+03-$	$2.45E+03$
F13	$3.75E+00\pm$	$1.13E+01\pm$	$6.00E+00\pm$	$4.03E+00\pm$	$8.90E+00\pm$	$2.61E+00\pm$	$2.96E+00\pm$
	$4.16E-01-$	$8.61E+00-$	$1.57E+00-$	$4.28E-01-$	$4.83E-01-$	$1.45E-01+$	$4.38E-01$
F14	$2.25E+01\pm$	$2.27E+01\pm$	$2.44E+01\pm$	$2.23E+01\pm$	$2.28E+01\pm$	$2.11E+01\pm$	$2.36E+01\pm$
	$2.06E-01+$	$1.81E-01+$	$3.08E-01-$	$3.79E-01+$	$3.26E-01+$	$5.41E-01+$	$2.62E-01$
F15	$8.87E+01\pm$	$3.84E+02\pm$	$3.97E+02\pm$	$2.26E+01\pm$	$2.60E+02\pm$	$3.52E+02\pm$	$3.20E+02\pm$
	$6.41E+01+$	$5.52E+01-$	$2.27E+02=$	$7.66E+01+$	$8.66E+01+$	$8.72E+01-$	$5.52E+01$
F16	$2.30E+02\pm$	$1.66E+02\pm$	$2.87E+02\pm$	$2.15E+02\pm$	$8.52E+01\pm$	$1.85E+01\pm$	$1.35E+02\pm$
	$4.41E+01-$	$9.88E+01-$	$2.48E+02-$	$2.50E+01-$	$3.88E+01+$	$1.34E+00+$	$5.19E+01$
F18	$9.46E+02\pm$	$9.24E+02\pm$	$9.12E+02\pm$	$9.65E+02\pm$	$9.17E+02\pm$	$9.13E+02\pm$	$9.56E+02\pm$
	$5.26E+00+$	$1.58E+00+$	$4.86E-01+$	$2.06E+01-$	$4.10E+00+$	$9.02E-01+$	$3.41E+01$
F19	$9.44E+02\pm$	$9.19E+02\pm$	$9.12E+02\pm$	$9.59E+02\pm$	$9.18E+02\pm$	$9.13E+02\pm$	$9.52E+02\pm$
	$6.29E+00+$	$2.49E+01+$	$4.87E-01+$	$1.33E+01-$	$3.14E+00+$	$1.91E+00+$	$3.31E+01$
F20	$9.44E+02\pm$	$9.24E+02\pm$	$9.12E+02\pm$	$9.63E+02\pm$	$9.17E+02\pm$	$9.14E+02\pm$	$9.58E+02\pm$
	$4.61E+00+$	$1.90E+00+$	$4.54E-01+$	$2.03E+01=$	$3.92E+00+$	$1.74E+00+$	$7.97E+00$
F21	$5.00E+02\pm$	$5.00E+02\pm$	$5.68E+02\pm$	$5.00E+02\pm$	$1.01E+03\pm$	$1.00E+03\pm$	$5.00E+02\pm$
	$7.54E-13-$	$2.11E-11-$	$1.98E+02-$	$1.99E-08-$	$2.11E+00-$	$1.41E+00-$	$3.28E-13$
F22	$1.00E+03\pm$	$9.69E+02\pm$	$8.58E+02\pm$	$1.09E+03\pm$	$8.98E+02\pm$	$8.75E+02\pm$	$9.93E+02\pm$
	$7.79E+00-$	$6.73E+00+$	$1.06E+01+$	$2.52E+01-$	$1.31E+01+$	$3.13E+00+$	$1.51E+01$
F23	$5.39E+02\pm$	$5.39E+02\pm$	$5.86E+02\pm$	$5.95E+02\pm$	$1.01E+03\pm$	$1.01E+03\pm$	$5.39E+02\pm$
	$8.67E-05+$	$2.75E-01-$	$1.52E+02-$	$7.69E+00-$	$1.87E+00-$	$1.41E+00-$	$1.62E-02$
−	12	16	11	15	12	8	/
+	7	5	7	3	9	12	/
=	2	0	3	3	0	1	/

"−," "+," and "=" denote that the performance of this algorithm is, respectively, worse than, better than, and similar to MDE according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

Table 4: Results obtained by the Multiple-Problem Wilcoxon test for twenty-one test functions at $D = 10$.

MDE versus	R^+	R^-	p value	At $\alpha = 0.05$	At $\alpha = 0.1$
CLPSO	169.0	41.0	0.016042	+	+
GL-25	184.0	47.0	0.016472	+	+
CMA-ES	192.0	18.0	0.001088	+	+
LBBO	141.0	90.0	0.366155	=	=
SFLSDE	147.0	63.0	0.112595	=	=
L-SHADE	136.5	73.5	0.232226	=	=

Table 5: Results obtained by the Multiple-Problem Wilcoxon test for twenty-one test functions at $D = 30$.

MDE versus	R^+	R^-	p value	At $\alpha = 0.05$	At $\alpha = 0.1$
CLPSO	200.0	31.0	0.003004	+	+
GL-25	198.5	32.5	0.003705	+	+
CMA-ES	177.0	54.0	0.031164	+	+
LBBO	160.5	70.5	0.113770	=	=
SFLSDE	184.0	47.0	0.016008	+	+
L-SHADE	116.5	114.5	0.95842	=	=

Table 6: Results obtained by the Multiple-Problem Wilcoxon test for twenty-one test functions at $D = 50$.

MDE versus	R^+	R^-	p value	At $\alpha = 0.05$	At $\alpha = 0.1$
CLPSO	155.5	75.5	0.157413	=	=
GL-25	179.5	51.5	0.024970	+	+
CMA-ES	136.0	95.0	0.465445	=	=
LBBO	175.5	55.5	0.035480	+	+
SFLSDE	138.0	93.0	0.424043	=	=
L-SHADE	94.0	116.0	1	=	=

Table 7: Average rankings of contraction criterion combinations by Friedman test at $D = 10$, $D = 30$, and $D = 50$.

$D = 10$		$D = 30$		$D = 50$	
Parameters	Ranking	Parameters	Ranking	Parameters	Ranking
$\rho_{1.0,1.0}$	4.5714	$\rho_{1.0,1.0}$	4.4286	$\rho_{1.0,1.0}$	**4.2619**
$\rho_{1.0,2.0}$	4.9286	$\rho_{1.0,2.0}$	4.881	$\rho_{1.0,2.0}$	4.4524
$\rho_{1.0,3.0}$	4.9524	$\rho_{1.0,3.0}$	4.381	$\rho_{1.0,3.0}$	5.0714
$\rho_{2.0,1.0}$	4.9762	$\rho_{2.0,1.0}$	4.9286	$\rho_{2.0,1.0}$	5
$\rho_{2.0,2.0}$	**3.5714**	$\rho_{2.0,2.0}$	**4.2381**	$\rho_{2.0,2.0}$	5.0476
$\rho_{2.0,3.0}$	6	$\rho_{2.0,3.0}$	5.6667	$\rho_{2.0,3.0}$	5.0238
$\rho_{3.0,1.0}$	5.1429	$\rho_{3.0,1.0}$	5.3571	$\rho_{3.0,1.0}$	5.2381
$\rho_{3.0,2.0}$	5.5714	$\rho_{3.0,2.0}$	5.5952	$\rho_{3.0,2.0}$	5.6905
$\rho_{3.0,3.0}$	5.2857	$\rho_{3.0,3.0}$	5.5238	$\rho_{3.0,3.0}$	5.2143

Table 8: Average rankings of $C_{\max}$ by Friedman test at $D = 10$, $D = 30$, and $D = 50$.

$D = 10$		$D = 30$		$D = 50$	
Parameters	Ranking	Parameters	Ranking	Parameters	Ranking
$C_{\max} = 3$	**3.4286**	$C_{\max} = 3$	3.5238	$C_{\max} = 3$	**3.6905**
$C_{\max} = 5$	3.5476	$C_{\max} = 5$	**3.3095**	$C_{\max} = 5$	4.2381
$C_{\max} = 7$	3.9762	$C_{\max} = 7$	4.0476	$C_{\max} = 7$	3.881
$C_{\max} = 9$	3.881	$C_{\max} = 9$	4.3333	$C_{\max} = 9$	3.8095
$C_{\max} = 11$	4.5238	$C_{\max} = 11$	4.2381	$C_{\max} = 11$	4.5238
$C_{\max} = 13$	3.7143	$C_{\max} = 13$	4.1429	$C_{\max} = 13$	4.119
$C_{\max} = 15$	4.9286	$C_{\max} = 15$	4.4048	$C_{\max} = 15$	3.7381

Table 9: Average rankings of M by Friedman test at $D = 10$, $D = 30$, and $D = 50$.

$D = 10$		$D = 30$		$D = 50$	
Parameters	Ranking	Parameters	Ranking	Parameters	Ranking
$M = 30$	**2.6429**	$M = 30$	**2.3095**	$M = 30$	3.1905
$M = 60$	2.7143	$M = 60$	2.6429	$M = 60$	2.7857
$M = 90$	2.8333	$M = 90$	2.9762	$M = 90$	**2.381**
$M = 120$	2.9762	$M = 120$	3.0238	$M = 120$	3.0952
$M = 150$	3.8333	$M = 150$	4.0476	$M = 150$	3.5476

Acknowledgments

The authors sincerely thank Dr. Simon D. for making the source code of LBBO available online, Dr. Hansen N. for making the source code of CMA-ES available online, and Dr. Ryoji Tanabe for making the source code of L-SHADE available online. This work was supported by the National Natural Science Foundation of China under Grant nos. 61103144, 61472375, and 41571403, by the China Postdoctoral Science Foundation under Grant no. 2012M511301, by Joint Funds of Equipment Pre-Research and Ministry of Education of China no. 6141A02022320, by the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) under Grant no. cug160207, and by the Laboratory Opening Fund of China University of Geosciences (Wuhan) under Grant no. SKJ2015222.

References

[1] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts—towards memetic algorithms," Technical Report Caltech Concurrent Computation Program 826, California Institute of Technology, Pasadena, Calif, USA, 1989.

[2] M. A. Ahandani, M.-T. Vakil-Baghmisheh, and M. Talebi, "Hybridizing local search algorithms for global optimization," *Computational Optimization and Applications*, vol. 59, no. 3, pp. 725–748, 2014.

[3] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera, "Memetic algorithms for continuous optimisation based on local search chains," *Evolutionary Computation*, vol. 18, no. 1, pp. 27–63, 2010.

[4] D. Simon, M. G. Omran, and M. Clerc, "Linearized biogeography-based optimization with re-initialization and local search," *Information Sciences*, vol. 267, pp. 140–157, 2014.

[5] J. Sun, J. M. Garibaldi, N. Krasnogor, and Q. Zhang, "An intelligent multi-restart memetic algorithm for box constrained global optimisation," *Evolutionary Computation*, vol. 21, no. 1, pp. 107–147, 2013.

[6] H. Wang, I. Moon, S. Yang, and D. Wang, "A memetic particle swarm optimization algorithm for multimodal optimization problems," *Information Sciences*, vol. 197, pp. 38–52, 2012.

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

[8] M. Di Carlo, M. Vasile, and E. Minisci, "Multi-population adaptive inflationary differential evolution," in *Proceedings of the Student Workshop on Bio-inspired Optimization Methods and their Applications (BIOMA '14)*, pp. 41–54, 2014.

[9] E. Minisci and M. Vasile, "Adaptive inflationary differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 1792–1799, July 2014.

[10] M. Vasile, E. Minisci, and M. Locatelli, "An inflationary differential evolution algorithm for space trajectory optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 267–281, 2011.

[11] A. Glotić and A. Zamuda, "Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution," *Applied Energy*, vol. 141, pp. 42–56, 2015.

[12] A. Zamuda and J. D. Hernández Sosa, "Differential evolution and underwater glider path planning applied to the short-term opportunistic sampling of dynamic mesoscale ocean structures," *Applied Soft Computing Journal*, vol. 24, pp. 95–108, 2014.

[13] L. Mingyong and C. Erbao, "An improved differential evolution algorithm for vehicle routing problem with simultaneous pick-ups and deliveries and time windows," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 2, pp. 188–195, 2010.

[14] M. Locatelli and M. Vasile, "(Non) convergence results for the differential evolution method," *Optimization Letters*, vol. 9, no. 3, pp. 413–425, 2015.

[15] L. dos Santos Coelho and V. C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 989–996, 2006.

[16] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[17] S. Muelas, A. LaTorre, and J.-M. Peña, "A memetic differential evolution algorithm for continuous optimization," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications (ISDA '09)*, pp. 1080–1084, December 2009.

[18] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Computing*, vol. 1, no. 2, pp. 153–171, 2009.

[19] Y. Wang, B. Li, and Z. He, "Enhancing differential evolution with effective evolutionary local search in memetic framework," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 2457–2464, IEEE, June 2011.

[20] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of*

the IEEE Congress on Evolutionary Computation (CEC '05), vol. 2, pp. 1785–1791, IEEE, September 2005.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] Y. Wang and B. Li, "A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization," in *Nature-Inspired Algorithms for Optimisation*, vol. 193 of *Studies in Computational Intelligence*, pp. 171–198, Springer, Berlin, Germany, 2009.

[23] M. Locatelli, M. Maischberger, and F. Schoen, "Differential evolution methods based on local searches," *Computers and Operations Research*, vol. 43, no. 1, pp. 169–180, 2014.

[24] M. Asafuddoula, T. Ray, and R. Sarker, "An adaptive hybrid differential evolution algorithm for single objective optimization," *Applied Mathematics and Computation*, vol. 231, pp. 601–618, 2014.

[25] A. K. Qin, K. Tang, H. Pan, and S. Xia, "Self-adaptive differential evolution with local search chains for real-parameter single-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 467–474, July 2014.

[26] A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl, "Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem," *Swarm and Evolutionary Computation*, vol. 23, pp. 50–64, 2015.

[27] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, W.-N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: a deep evolutionary algorithm," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1798–1810, 2015.

[28] X. Sun and Z.-Q. Wu, "Fractal and multifractal description of surface topography," *Acta Physica Sinica*, vol. 50, no. 11, pp. 2130–2131, 2001.

[29] R. Ros, "Benchmarking the bfgs algorithm on the bbob-2009 function testbed," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pp. 2409–2414, 2009.

[30] A. Zamuda, J. Brest, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 1925–1931, IEEE, June 2013.

[31] P. N. Suganthan, "Problem definitions and evaluation criteria for the cec2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.

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

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

[34] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 1658–1665, IEEE, July 2014.

[35] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2009.

[36] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[37] J. Alcalá-Fdez, L. Sánchez, S. García et al., "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.