Information Sciences 181 (2011) 3175–3187

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An effective memetic differential evolution algorithm based on chaotic local search

Dongli Jia [a,b,*], Guoxin Zheng [a], Muhammad Khurram Khan [c]

[a] Key Laboratory of Special Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200072, China
[b] School of Information and Electronic Engineering, Hebei University of Engineering, Handan 056038, China
[c] Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 11653, Saudi Arabia

A R T I C L E   I N F O

Article history:
Received 30 April 2010
Received in revised form 15 March 2011
Accepted 25 March 2011
Available online 3 April 2011

A B S T R A C T

This paper proposes an effective memetic differential evolution (DE) algorithm, or DECLS, that utilizes a chaotic local search (CLS) with a 'shrinking' strategy. The CLS helps to improve the optimizing performance of the canonical DE by exploring a huge search space in the early run phase to avoid premature convergence, and exploiting a small region in the later run phase to refine the final solutions. Moreover, the parameter settings of the DECLS are controlled in an adaptive manner to further enhance the search ability. To evaluate the effectiveness and efficiency of the proposed DECLS algorithm, we compared it with four state-of-the-art DE variants and the IPOP-CMA-ES algorithm on a set of 20 selected benchmark functions. Results show that the DECLS is significantly better than, or at least comparable to, the other optimizers in terms of convergence performance and solution accuracy. Besides, the DECLS has also shown certain advantages in solving high dimensional problems.

1. Introduction

Differential evolution (DE), which was first proposed over 1994–1996 by Storn and Price at Berkeley, is a simple yet powerful evolutionary algorithm [4,23,25,26]. It has been proved that DE is an accurate, reasonably fast, and robust optimizer for many optimization problems in real-world applications [6,7,11,22,34,35,30]. However, the DE does not guarantee the convergence to the global optimum. It is easily trapped into local optima resulting in a low optimizing precision or even failure.

There are two main ways to enhance the optimizing performance of DE.

(1) Adaptively controlling the parameters of DE within the whole evolution process. There are mainly three parameters for DE, namely, Scaling factor *F*, Cross Rate factor *CR*, and Population Size *NP*. All of them are closely related to the convergence performance. Generally, small *F* and *CR* values benefit to a fast convergence speed, while large *F* and *CR* values help to increase the diversity of population, which is good for avoiding premature convergence. The *NP* is also concerned with the diversity of the population, and it is more closely associated with the dimension of the problems being solved. To get a balance between the convergence speed and premature convergence, a couple of adaptive DE algorithms have been proposed. A typical example is the SaDE algorithm, proposed by Qin and Suganthan in [24]. It can automatically adapt its learning strategies and the parameter settings during the evolutionary procedure. Results show that SaDE is better than, or at least comparable to, the classic DE and its variants in terms of convergence

* Corresponding author at: Key Laboratory of Special Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200072, China.
E-mail address: jwdsli@163.com (D. Jia).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.03.018

performance for a set of 25 benchmark functions provided by the CEC2005 special session on real parameter optimization. Other adaptive DE algorithms such as fuzzy adaptive DE (FADE), adaptive DE with optional archive (JADE), and jDE can be found in the literature [2,3,16,36], respectively.

(2) Incorporating local search (LS) into DE. DE has a good global exploration performance but suffers from stagnation problems [15]. On the contrary, LS can compensate for this deficiency by refining individuals with a neighborhood search procedure. A DE embedded with LS operation is generally called a memetic DE algorithm. In [21], Noman and Iba proposed a crossover-based adaptive local search (LS) for enhancing the performance of standard DE. Simulations showed that the proposed algorithm outperforms the classic DE in terms of convergence velocity in all experimental studies. Other memetic DE algorithms such as memetic DE to deal with continuous problems (MDE-DC) and generalized opposition-based learning DE (GODE) can be found in the literature [20,32].

While there are already a lot of suggestions as discussed above for enhancing the performance of DE, the optimizing results for some benchmark functions are still unacceptable. The DE must be further improved to achieve better solutions to different kind of problems.

In this paper, an effective memetic DE algorithm based on chaos local search (DECLS) is originally proposed. Chaotic search usually works well in local optimization for its ergodicity and randomicity [29,31]. However, it deteriorates its performance when exploring a large search space. To overcome this difficulty, a shrinking strategy for the search space of the CLS is introduced. Besides, to save the function evaluations (FEs), the local search length is dynamically set based on the feedback of the fitness of the objective functions. Moreover, the parameter settings of the DECLS algorithm are also adapted in the evolution process to further improve the optimization efficiency. The combination of the DE with a CLS and a parameter adaptation mechanism is very reasonable. The CLS helps to enhance the local search ability of DE, while the parameter adaptation improves the global optimizing quality. The new proposed DECLS algorithm was compared with four state-of-the-art DE variants and the IPOP-CMA-ES algorithm on a set of 20 selected benchmark functions taken from [33] and CEC'05. Results show that not only our algorithm is significantly better than the classic DE, but it is a promising method for solving high dimensional optimization problems.

The remainder of this paper is organized as follows. Section 2 describes the classic DE procedure and some basic concepts. In Section 3, the new proposed DE algorithm with chaotic local search is elaborated with detailed explanations. Simulations are presented in Section 4 for the comparison and analysis. At the end, Section 5 concludes the findings in the paper.

2. Classic differential evolution

Similar to other population based optimization algorithms, DE also involves two phases: initialization and evolution. In the initialization phase, the DE population is generated randomly if nothing is known about the problem. In the evolution phase, individuals from the population go through mutation, crossover, and selection process repeatedly until the termination criterion is met.

Without loss of generality, a minimization problem $f(X)$ is discussed here.

$$\begin{aligned} \min \quad & f(X), \quad X = [x_1, x_2, \ldots, x_n], \\ \text{s.t.} \quad & x_i \in [a_i, b_i], \end{aligned} \tag{1}$$

where $f(X)$ is the objective function, and X is the decision vector consisting of n variables. a_i and b_i indicate the lower and upper boundaries of x_i, respectively.

In the standard DE model, each individual represents a candidate solution to $f(X)$ within the search space. At each generation, the objective function is evaluated for each individual. The obtained value, or fitness, is used to assess the quality of the individuals and the best member is noted in order to keep track of the progress that is made during the optimization process.

2.1. Initialization

The individuals of DE take the form as follows:

$$X_i^g = [x_{i1}^g, x_{i2}^g, \ldots, x_{iD}^g], \quad i \in \{1, 2, \ldots, NP\} \tag{2}$$

where x_{ij}^g denotes the jth candidate parameter from i individual in gth generation for the objective function $f(X)$. NP is the population size. D is the dimension of objective function. In the initialization phase, all the individuals from the whole population are initialized randomly with uniform probability distribution in its search space unless otherwise stated.

2.2. Mutation

The primary difference between the DE and genetic algorithm (GA) is in mutation. There are different mutation methods in DE. The simplest method is described below. For each individual vector X_i^g, DE generates a mutated vector V_i^g based on the difference among the randomly selected individuals.

$$V_i^g = X_{r3}^g + F^k \cdot (X_{r2}^{gk} - X_{r1}^{gk}),$$
$$r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \quad k \in \{1, 2\}, \quad F^k \in [0, 1], \tag{3}$$

where F^k is called scaling factor which is closely related to the convergence speed. k indicates the number of difference vectors taking part in the mutation operation. In practice, F^1 usually equals to F^2 when there are two difference vectors. X_{r3}^g, X_{r2}^g and X_{r1}^g are selected individuals from the population but different from the running individual X_i^g.

2.3. Crossover

In this crossover scheme, the candidate child vector is generated by the following equation:

$$u_{ij}^g = \begin{cases} v_{ij}^g, & rand(j) \leqslant CR, \\ x_{ij}^g, & rand(j) > CR, \end{cases} \quad j \in \{1, 2, \ldots, D\} \tag{4}$$

where u_{ij}^g is a parameter of candidate child individual U_i^g, and the v_{ij}^g is a parameter of V_i^g. CR is called crossover factor which limited to [0, 1]. There are primarily two different crossover schemes called binomial crossover and exponential crossover. The difference between them lies in the way the vector U_i^g generates. In classic DE, CR is a fixed value used for all individuals. In adaptive DE algorithms, CR usually varies with the evolution process.

2.4. Selection

Selection takes the competition mechanism. The candidate child U_i^g and the old individual X_i^g compete in their fitness. The winner will have the chance to survive for the next generation.

$$X_i^{g+1} = \begin{cases} U_i^g, & \text{if } f(X_i^g) > f(U_i^g), \\ X_i^g, & \text{otherwise}. \end{cases} \tag{5}$$

The DE repeats above B, C and D process until the termination condition is met and outputs a final candidate solution to $f(X)$.

There are several variants of DE based on different mutation and crossover schemes. This can be defined as DE/x/y/z. Where x specifies the individual to be mutated, y is the number of difference individuals used, and z denotes the crossover scheme. In our paper, DE only denotes the most typical DE/rand/1/bin strategy (if no other strategy is specified).

3. Differential evolution with chaotic local search and parameter adaptation

Chaos is a typical nonlinear phenomenon in nature which is characterized by ergodicity, randomicity and sensitivity to its initial conditions [14]. Because of the Ergodicity and randomicity, a chaotic system changes randomly, but eventually goes through every state if the time duration is long enough. This characteristic of chaotic systems can be utilized to build up a search operator for optimizing objective functions. However, Chaos optimization works well in a small search space but generates unacceptable optimization time in a large search space. Therefore, chaotic search is often incorporated into other global optimizers such as GAs and Particle Swarm Optimization (PSO) to enhance their search ability [12,17,18].

3.1. Chaotic iteration

In this paper, the Logistic chaotic function is employed to construct a chaotic DE as follows:

$$\beta_j^{k+1} = \mu\beta_j^k(1 - \beta_j^k), \quad k = 1, 2, \ldots; \ \beta_j \in (0, 1), \ \beta_j \neq 0.25, 0.5, \text{ and } 0.75, \tag{6}$$

where β_j^k is the jth chaotic variable in kth generation. When $\mu = 4$, the Logistic function comes into a thorough chaotic state. Fig. 1 shows the ergodic property and the probability distribution of the Logistic function when $\beta_j^1 = 0.11231$ and iterations = 1000. It can be seen from the Fig. 1 that the chaotic iteration has a higher probability searching in boundary fields which is in turn a benefit as an operator to avoid premature convergence to some extent [13,36].

3.2. Chaotic search with "shrinking" strategy

A local search is usually employed to refine the solutions of objective functions. However, it may lead to unacceptably rapid convergence hence resulting in the whole evolution process trapped into local optima. The advantage of CLS just lies in its randomicity which helps avoid premature convergence in the search process. To enhance the exploitation abilities of CLS and refine the solutions in the later phase, we shrink the search space of the CLS with the growth of the function evaluations.

The formula of chaotic local search is:

$$X_i'^{(g)} = (1 - \lambda)X_i^{(g)} + \lambda\beta_c \tag{7}$$

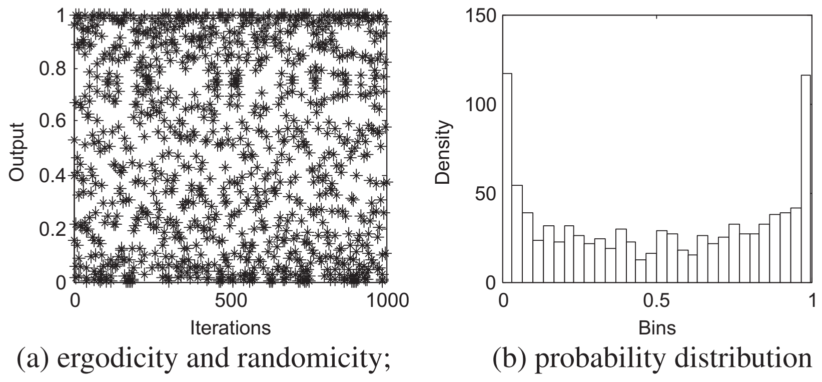


Fig. 1. Property of logistic function.

where $X_i'^{(g)}$ is a new vector of individual X_i^g in gth generation produced by chaotic local search. β_c is generated by the equation $\beta_c = A + \beta_j^k \cdot (B - A)$, where β_j^k is obtained from chaotic iteration (β_j^1 is randomly selected within $[0,1]$) and mapped into the search space $[A,B]$of individual X_i. λ is the shrinking scale given by:

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

where *FEs* are the current function evaluations. And *m* controls the shrinking speed. With higher *m* values, the shrinking speed is slower.

3.3. Search length of CLS

A proper search length is very important for CLS. A small length may be inefficient in exploring the best solution and therefore unsuccessful at improving the search quality. On the other hand, with a longer length, the CLS may consume additional function evaluations unnecessarily. Considering the function complexities and the allowed max function evaluations, we set the maximum search length of the CLS as: $S_l = \frac{D}{5}$, where, S_l is the search length of CLS.

In addition, to avoid premature convergence and save FEs, CLS stops once a better fitness is obtained.

3.4. Controlling the parameters of DE in an adaptive manner

In the DECLS algorithms, a simple parameter adaptation method is implemented. First, each individual of the population is independently assigned an *F* and a *CR* within the range of $[0,1]$. Then in every generation, one percent of uniformly selected individuals of the population randomly change their *F* and *CR* values within the ranges $[0.1,0.9]$ and $[0,1]$, respectively. In the selection phase, the winner of the individuals will survive and their parameters of *F* and *CR* will propagate to the next generation.

3.5. Improved DE with CLS

For the successful incorporation of a CLS in DE, the individuals that will undergo the CLS process must be properly selected. To reduce function evaluations, only the best individual of the whole population is selected for using the CLS. This is very important because applying CLS on an ordinary individual may progress slowly and unnecessarily waste function evaluations.

The whole procedure of the DECLS is detailed below:

```
Step 1. Initialization
    Step 1.1. Set the population size NP, and max FEs;
    Step 1.2. Initialize all individuals $X_i^0$ with independent F and CR within search space;
    Step 1.3. Evaluate f(X) over all individuals.
Step 2. Iteration
    Step 2.1. Execute mutation according to Eq. (3) and adjusting F and CR;
    Step 2.2. Execute crossover according to Eq. (4);
    Step 2.3. Execute selection according to Eq. (5);
Step 3. Apply CLS on the best individual obtained so far.
Step 4. If the stopping criterion is met, output the best solution. Otherwise, jump to step 2.
```

4. Experiments

4.1. Test problems and the performance evaluation criteria

Experiments were carried out over a set of 20 widely used benchmark functions with different characteristics. Among these benchmarks, functions F1-F13 described in Appendix A are traditional scalable functions taken from [33] by Yao et al. F1–F4 are unimodal functions, F5 is the Rosenbrock function which is unimodal function for $D = 2$ and multimodal function for $D > 3$[28], F6 is a step function, F7 is a noisy quartic function, F8–F13 are multimodal functions. Functions F14—F20 are taken from CEC'05 test problems [27]. F14–F18 are shifted and rotated functions and F19–F20 are expanded functions which are listed as F6–F10 and F13–F14 in [27], respectively. All these functions taken from the CEC2005 special session on real parameter optimization were designed to avoid the behavior of some algorithms that especially benefit from the functions whose minima lie in the origin.

Because of these shifted functions, we choose the function error value to evaluate the performance of the compared algorithms. The function error value for a solution x is defined as $(f(x) - f(x^*))$, where x^* is the global optimum of the objective function[21].

4.2. Effects of CLS on DE

In order to demonstrate the effects of CSL on improving the performance of DE, we compared the DECLS algorithm with the canonical DE on the test problems at the dimensions $D = 25$ and $D = 100$. Additionally, the DECLS without CLS was also simulated to show the effects of our parameter adaptation strategy.

The parameters of the canonical DE were set to be $F = 0.5$, $CR = 0.9$, population size $NP = 10\text{D}$ as recommended in [2,19] and [20]. For DECLS, the original parameters were set to be $F = 0.5$, $CR = 0.5$, $NP = D$, and the CLS search length $L = D/5$ as discussed in Section 3. After some experiments, we chose $m = 1500$ for the comparisons (which is discussed in Section 4.4). The DECLS without CSL used the same parameter settings as DECLS except that the CLS search was disabled. Finally, the maximum FEs was set to be 8000D for all the algorithms.

The averaged and standard deviations of the best error values for 25 independent runs of each algorithm are presented in Table 1. The best solutions and its standard deviation have been shown in boldface.

From Table 1, it can be seen that the performance of the DECLS without CLS is superior to that of the DE except for the result of function F3 at $D = 25$. Though the parameter adaptation can improve the performance of DE effectively, the CLS is still needed. In 36 out of 40 cases in Table 1, the DECLS achieves the best error values. Especially for a high dimension $D = 100$, there is a remarkable improvement in the quality of the solutions from DECLS. So it can be concluded that the effects of CLS on improving the DE are very significant.

It should be also noted that, in some cases, the benefit from CLS search may not be enough to compensate for the consumption of function evaluations. For instance, for the functions F8, F11, F12 and F13 at $D = 25$, the local search has deteriorated the results of the DECLS without CLS algorithm.

4.3. Compared DECLS with four state-of-the-art DE variants and the IPOP-CMA-ES algorithm

4.3.1. Parameter settings for the competitors

To show the superiority of the proposed DECLS algorithm, we compared it with four state-of-the-art DE variants, namely, SaDE, DEGL[8], jDE, and DEahcSPX[21]. The IPOP-CMA-ES, which is the best optimizer in CEC'05 special session on real parameter optimization, was also compared [1]. The comparisons were carried out on the test suit at dimensions $D = 25$ and $D = 100$. Since the performance of evolutionary algorithms strongly depends on their parameter settings, for fair comparisons, all algorithms in this study directly take their parameters from the relevant literature.

The parameters of jDE were set to be $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, $F_u = 0.9$ and $NP = 10D$ as it used in [2]. The parameters of DEahcSPX were set to be $F = 0.9$, $CR = 0.9$, $np = 3$ and $NP = D$ as in [21]. For SaDE, the population size NP was set to $10D$ as classic DE, F was initialized to different random values in the range $(0,2]$ with normal distributions of mean 0.5 and standard deviation 0.3, and CR was normally distributed with mean 0.5 and standard deviation 0.1 for different individuals as in [24]. For DEGL, the parameters were set to be: $\alpha = \beta = F = 0.8$, $CR = 0.9$, $NP = 10D$ and the neighborhood size $k = NP/10$ as recommended in [8]. The authors also provide four variants of DEGL, namely, DEGL/SAW, DEGL/Li, DEGL/EI, and DEGL/RandW. In our experiments, only the DEGL/SAW was tested for comparison since it has shown the best performance in the literature. The parameters of the IPOP-CMA-ES algorithm were chose as prescribed in [1]. In all comparisons, the maximum FEs was set to be 8000D.

4.3.2. Comparisons of the final solutions accuracy

The average and standard deviations (within parentheses) of the best error values of 25 independent runs for each algorithm have been summarized in Tables 2 and 3 for the dimensions $D = 25$ and $D = 100$, respectively. The best solutions and their standard deviations have been shown in boldface.

Table 1
Best error values at D = 25 and D = 100.

	D	DE	DECLS without CLS	DECLS
F1	25	9.59E−05 ± 3.10E−05	8.30E−14 ± 2.71E−14	**7.01E−134 ± 1.53E−133**
	100	1.12E+04 ± 8.96E+02	5.84E−03 ± 9.22E−04	**5.28E−61 ± 5.64E−61**
F2	25	8.91E−03 ± 1.68E−03	6.10E−09 ± 1.46E−09	**3.39E−80 ± 5.81E−80**
	100	1.79E+02 ± 5.19E+01	2.27E−02 ± 2.45E−03	**3.65E−36 ± 2.01E−36**
F3	25	1.31E+02 ± 3.24E+01	3.84E+02 ± 1.70E+02	**2.30E−08 ± 6.01E−08**
	100	1.53E+05 ± 1.65E+04	1.19E+05 ± 7.09E+03	**1.40E+02 ± 4.94E+01**
F4	25	7.36E−01 ± 1.96E−01	6.89E−02 ± 1.67E−02	**9.93E−05 ± 2.05E−04**
	100	6.05E+01 ± 3.13E+00	2.36E+01 ± 5.02E−01	**3.56E−02 ± 3.34E−02**
F5	25	1.91E+01 ± 3.19E−01	1.86E+01 ± 4.2E−01	**8.86E−01 ± 1.76E+00**
	100	6.58E+06 ± 1.67E+06	1.67E+02 ± 5.35E+01	**9.80E+01 ± 4.48E+01**
F6	25	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	**0.00E+00 ± 0.00E+00**
	100	1.15E+04 ± 1.58E+03	0.00E+00 ± 0.00E+00	**0.00E+00 ± 0.00E+00**
F7	25	1.61E−02 ± 2.59E−03	8.92E−03 ± 2.63E−03	**2.31E−03 ± 9.79E−04**
	100	8.98E+00 ± 2.85E+00	7.75E−02 ± 1.29E−02	**8.09E−03 ± 1.51E−03**
F8	25	6.09E+03 ± 1.73E+02	**1.74E−05 ± 1.34E−05**	7.89E+01 ± 8.37E+01
	100	3.32E+04 ± 4.52E+02	1.18E+04 ± 7.61E+02	**9.46E−11 ± 0.00E+00**
F9	25	1.39E+02 ± 1.29E+01	2.56E+00 ± 8.10E−01	**0.00E+00 ± 0.00E+00**
	100	9.88E+02 ± 1.84E+01	2.75E+02 ± 1.96E+01	**0.00E+00 ± 0.00E+00**
F10	25	3.68E−03 ± 7.86E−04	7.59E−08 ± 1.55E−08	**3.55E−15 ± 1.03E−54**
	100	1.17E+01 ± 4.43E−01	1.03E−02 ± 9.12E−04	**7.11E−15 ± 0.00E+00**
F11	25	7.36E−02 ± 1.01E−01	**2.94E−12 ± 3.49E−12**	5.71E−03 ± 1.71E−02
	100	1.10E+02 ± 1.13E+01	3.98E−03 ± 3.82E−04	**0.00E+00 ± 0.00E+00**
F12	25	2.56E−05 ± 9.05E−06	9.11E−15 ± 6.36E−15	2.64E−07 ± 7.93E−07
	100	2.45E+06 ± 9.66E+05	1.89E−03 ± 5.97E−04	**4.71E−33 ± 7.26E−49**
F13	25	1.14E−04 ± 4.64E−05	**4.06E−14 ± 1.95E−14**	5.07E−07 ± 1.62E−06
	100	1.38E+07 ± 4.07E+06	1.95E−02 ± 3.16E−03	**1.35E−32 ± 0.00E−00**
SF1	25	2.45E+01 ± 9.61E−01	5.02E+01 ± 3.45E+01	**9.14E+00 ± 2.15E+01**
	100	3.82E+09 ± 9.94E+08	5.68E+02 ± 5.80E+01	**9.59E+01 ± 2.58E+01**
SF2	25	4.05E+03 ± 1.02E+03	3.94E+03 ± 1.20E+03	**3.15E+03 ± 6.71E+02**
	100	1.30E+04 ± 1.45E+03	1.25E+04 ± 1.19E+03	**1.18E+04 ± 1.48E+03**
SF3	25	2.09E+01 ± 3.95E−02	2.09E+01 ± 1.05E−01	**2.05E+01 ± 1.27E−01**
	100	2.13E+01 ± 1.20E−02	2.13E+01 ± 3.43E−02	**2.05E+01 ± 4.94E−02**
SF4	25	1.40E+02 ± 1.42E+01	1.87E+00 ± 3.22E−01	**0.00E+00 ± 0.00E+00**
	100	1.10E+03 ± 2.37E+01	2.68E+02 ± 2.36E+01	**5.68E−14 ± 0.00E+00**
SF5	25	1.62E+02 ± 9.65E+00	1.09E+02 ± 1.11E+01	**4.65E+01 ± 1.10E+01**
	100	1.23E+03 ± 3.47E+01	8.21E+02 ± 3.05E+01	**3.09E+02 ± 6.23E+01**
EF1	25	1.34E+01 ± 9.46E−01	3.46E+00 ± 3.63E−01	**1.04E+00 ± 2.39E−01**
	100	1.32E+02 ± 8.07E+00	4.06E+01 ± 1.70E+00	**8.16E+00 ± 1.21E+00**
EF2	25	1.08E+01 ± 1.64E−01	1.07E+01 ± 1.33E−01	**1.01E+01 ± 3.84E−01**
	100	4.76E+01 ± 1.60E−01	4.73E+01 ± 1.49E−01	**4.58E+00 ± 6.56E−01**

From the Tables 2 and 3, it can be seen that the performance of the proposed DECLS algorithm is superior to that of the four state-of-the-art DE variants as well as the IPOP-CMA-ES algorithm. One may note that the DE variants and the IPOP-CMA-ES algorithm also perform well at dimension D = 25, but the DECLS still remains the most effective algorithm. In 10 out of 20 cases, the DECLS achieved the best average accuracy. Table 3 further indicates that the DECLS has certain advantages on optimizing high dimensional problems. In Table 3, the DECLS outperforms all the other DE variants over all benchmark functions, and outperforms the IPOP-CMA-ES in 15 out of the 20 test problems.

4.3.3. *Wilcoxon test*

To further confirm above statement, a non-parametric test, Wilcoxon two-side signed rank test, was employed to perform pair-wise comparisons between two algorithms [10]. The p-values obtained considering all the function results (20 values) for each dimension are summarized in Table 4.

From the Table 4, it can be seen that the DECLS algorithm really outperforms the four DE variants with a level of significance α = 0.05 considering independent pair-wise comparisons. However, we can only say that the DECLS is comparable to the IPOP-CMA-ES algorithm, for the p-value is too high and not significant.

Table 2
Averaged best error values and its standard deviation (within parentheses) at $D = 25$.

	DEGL/saw	jDE	DEahcSPX	SaDE	IPOP-CMA-ES	DECSL
F1	1.61E−07 (4.83E−07)	1.67E−08 (4.81E−09)	2.25E−16 (2.31E−16)	1.61E−24 (1.54E−24)	1.47E−15 5.76E−16	**7.01E−134** (**1.53E−133**)
F2	1.31E−03 (2.92E−03)	8.37E−06 (1.88E−06)	5.52E−10 (3.83E−10)	7.19E−13 (2.88E−13)	2.58E−11 5.85E−12	**3.39E−80** (**5.81E−80**)
F3	1.33E−02 (2.61E−02)	9.58E+02 (1.74E+02)	1.73E−04 (2.41E−04)	7.28E−08 (6.01E−08)	2.79E+05 9.99E+04	**2.30E−08** (**6.01E−08**)
F4	6.10E−02 (9.79E−02)	3.37E−03 (4.87E−02)	9.45E−01 (7.73E−01)	4.93E−05 (1.77E−05)	**3.52E−11** **1.41E−11**	9.93E−05 (2.05E−04)
F5	3.01E+00 (5.89E+00)	1.95E+01 (1.63E−01)	4.22E+00 (3.34E+00)	1.16E+01 (2.57E+00)	**2.89E−15** **1.13E−15**	8.86E−01 (1.76E+00)
F6	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**)
F7	4.58E−03 (4.84E−03)	1.21E−02 (2.46E−03)	**9.13E−04** (**2.52E−04**)	3.86E−03 (1.38E−03)	5.68E−2 1.98E−2	2.31E−03 (9.79E−04)
F8	5.87E+03 (3.22E+02)	**2.80E+01** (**1.98E+01**)	7.23E+02 (4.78E+02)	1.13E+03 (3.18E+02)	5.52E+03 3.47E+03	7.89E+01 (8.37E+01)
F9	1.20E+02 (1.17E+01)	3.58E+01 (5.22E+00)	2.15E+01 (1.13E+01)	3.16E+01 (2.71E+00)	1.77E+00 1.30E+00	**0.00E+00** (**0.00E+00**)
F10	6.58E−12 (1.97E−12)	3.13E−05 (4.54E−06)	3.99E−10 (2.87E−10)	4.29E−13 (2.76E−13)	2.05E+01 4.14E−01	**3.55E−15** (**1.03E−54**)
F11	6.02E−03 (9.87E−03)	3.10E−07 (2.60E−07)	7.11E−04 (1.20E−03)	**0.00E+00** (**0.00E+00**)	1.97E−15 9.20E−16	5.71E−03 (1.71E−02)
F12	1.81E−06 (5.42E−06)	9.64E−10 (2.35E−10)	3.28E−17 (4.45E−17)	**1.98E−25** (**2.41E−25**)	3.02E−15 7.73E−16	2.64E−07 (7.93E−07)
F13	1.76E−06 (5.25E−05)	9.25E−09 (4.78E−09)	2.44E−04 (4.84E−04)	**8.33E−25** (**7.12E−25**)	2.84E−15 7.14E−16	5.07E−07 (1.62E−06)
SF1	3.42E+00 (8.20E+00)	4.59E+01 (3.81E+01)	1.51E+01 (1.93E+01)	5.70E+01 (2.95E+01)	**1.03E−24** **2.01E−14**	9.14E+00 (2.15E+01)
SF2	3.20E+03 (6.81E+02)	3.00E+03 (9.01E+02)	3.87E+03 (1.06E+03)	3.59E+03 (7.51E+02)	**0.00E+00** (**0.00E+00**)	3.15E+03 (6.71E+02)
SF3	2.09E+01 (4.52E−02)	2.08E+01 (6.29E−02)	2.09E+01 (5.11E−02)	2.09E+01 (8.48E−02)	3.046E+01 (4.52E−01)	**2.05E+01** (**1.27E−01**)
SF4	1.31E+02 (1.57E+01)	1.54E+01 (1.65E+01)	1.89E+01 (5.95E+00)	2.96E+01 (2.88E+00)	2.98E+00 (1.49E+00)	**0.00E+00** (**0.00E+00**)
SF5	1.40E+02 (1.51E+01)	1.13E+02 (1.29E+01)	7.91E+01 (6.64E+01)	1.01E+02 (1.20E+01)	**1.82E+00** (**8.58E−01**)	4.65E+01 (1.10E+01)
EF1	1.15E+01 (9.05E−01)	4.62E+00 (1.44E+00)	2.20E+00 (5.30E−01)	5.47E+00 (4.17E−01)	1.96E+00 (2.25E−01)	**1.04E+00** (**2.39E−01**)
EF2	1.05E+01 (2.29E−01)	1.05E+01 (2.04E−01)	1.07E+01 (4.16E−01)	1.07E+01 (1.92E−01)	1.07E+01 (8.30E−01)	**1.01E+01** (**3.84E−01**)

When considering multiple comparisons [10], the p-values are

$$\begin{aligned} p &= 1 - ((1 - 9.6733\mathrm{E} - 04) \cdot (1 - 4.8625E - 02) \cdot (1 - 2.2252E - 03) \cdot (1 - 6.2103E - 03) \cdot (1 - 3.7589E - 01)) \\ &= 4.1181\mathrm{E} - 01 \end{aligned}$$

and

$$\begin{aligned} p &= 1 - ((1 - 8.8575\mathrm{E} - 05) \cdot (1 - 8.8575\mathrm{E} - 05) \cdot (1 - 1.3183E - 04) \cdot (1 - 1.3183E - 04) \cdot (1 - 2.7724\mathrm{E} - 01)) \\ &= 2.7756\mathrm{E} - 01 \end{aligned}$$

for the dimensions $D = 25$ and $D = 100$, respectively. Since the DECLS is not obviously better against the CMA-ES algorithm, the p-values obtained from the multiple comparisons are not significant. However, when only performing multiple comparisons over the remaining DE variants, we can safely confirm that the DECLS is superior to the other DE variants with the p-value of $p = 5.7550\mathrm{E}{-02}$ for the dimension $D = 25$ and $p = 4.4074\mathrm{E}{-04}$ for the dimension $D = 100$. Moreover, the Wilcoxon test also confirms that the DECLS algorithm is more appropriate for high dimensional problems.

Table 3
Averaged best error values and its standard deviation (within parentheses) at $D = 100$.

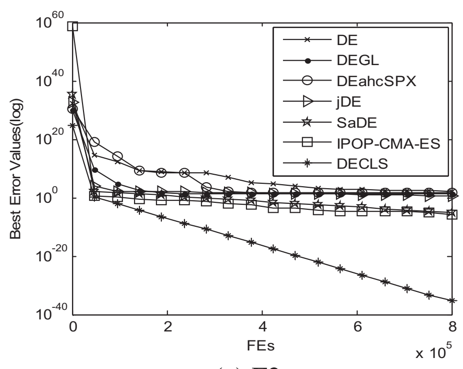
	DEGL/saw	jDE	DEahcSPX	SaDE	IPOP-CMA-ES	DECLS
F1	5.36E+02	5.63E+01	2.75E+03	3.11E−11	7.91E−16	**5.28E−61**
	(1.07E+03)	(2.67E+00)	(3.15E+03)	(6.95E−12)	1.88E−16	(**5.64E−61**)
F2	1.48E+01	4.17E+00	3.51E+01	6.31E−06	1.63E−06	**3.65E−36**
	(3.43E+01)	(2.91E−01)	(3.95E+01)	(1.37E−06)	4.56E−06	(**2.01E−36**)
F3	2.25E+04	1.23E+05	3.62E+04	1.84E+02	1.26E+06	**1.40E+02**
	(2.58E+04)	(1.82E+04)	(3.68E+04)	(6.88E+01)	1.28E+05	(**4.94E+01**)
F4	6.35E+00	4.07E+01	2.87E+01	6.21E+00	**7.37E−11**	3.56E−02
	(1.31E+01)	(7.42E−01)	(1.47E+01)	(9.92E−01)	**9.39E−12**	(3.34E−02)
F5	2.93E+04	1.14E+04	1.56E+06	1.21E+02	**1.66E−14**	9.80E+01
	(8.76E+04)	(1.46E+03)	(1.90E+06)	(3.79E+01)	**4.96E−15**	(4.48E+01)
F6	2.56E+02	6.91E+01	2.67E+03	0.00E+00	0.00E+00	**0.00E+00**
	(7.11E+02)	(3.85E+00)	(2.99E+03)	(0.00E+00)	(0.00E+00)	(**0.00E+00**)
F7	8.56E−01	3.01E−01	1.91E+00	2.12E−02	2.59E−01	**8.09E−03**
	(1.65E+00)	(5.07E−02)	(1.18E+00)	(4.49E−03)	2.33E−02	(**1.51E−03**)
F8	3.30E+04	2.27E+04	2.62E+04	2.31E+04	1.21E+04	**9.46E−11**
	(6.16E+02)	(7.06E+02)	(6.63E+03)	(5.55E+02)	3.31E+03	(**0.00E+00**)
F9	8.45E+02	5.99E+02	5.46E+02	4.82E+02	5.53E+00	**0.00E+00**
	(6.59E+01)	(3.56E+01)	(3.91E+02)	(1.57E+01)	3.07E+00	(**0.00E+00**)
F10	2.71E+00	2.38E+00	4.39E+00	8.23E−07	2.09E+01	**7.11E−15**
	(3.23E+00)	(5.10E−02)	(4.78E+00)	(1.12E−07)	5.28E−01	(**0.00E+00**)
F11	1.30E+01	1.54E+00	2.49E+01	2.74E−03	1.11E−14	**0.00E+00**
	(2.94E+01)	(2.26E−02)	(2.78E+01)	(5.96E−03)	2.39E−15	(**0.00E+00**)
F12	7.25E+01	1.22E+01	1.71E+05	3.28E−12	7.48E−15	**4.71E−33**
	(2.17E+02)	(1.28E+00)	(2.86E+05)	(5.28E−12)	2.27E−15	(**7.26E−49**)
F13	3.77E+05	1.12E+02	1.89E+06	2.11E−12	6.70E−15	**1.35E−32**
	(1.13E+06)	(1.54E+01)	(2.19E+06)	(1.27E−11)	2.11E−15	(**0.00E−00**)
SF1	1.61E+08	5.50E+02	2.38E+09	1.56E+02	**8.86E−01**	9.59E+01
	(3.22E+08)	(9.00E+01)	(7.25E+08)	(5.01E+01)	(**1.76E+00**)	(2.58E+01)
SF2	1.27E+04	1.26E+04	1.18E+04	1.26E+04	**2.84E−14**	1.18E+04
	(1.13E+03)	(1.77E+03)	(1.30E+03)	(1.72E+03)	(**1.01E−14**)	(1.48E+03)
SF3	2.13E+01	2.13E+01	2.13E+01	2.13E+01	2.13E+01	**2.05E+01**
	(2.16E−02)	(2.07E−02)	(2.95E−02)	(2.40E−02)	(1.87E−02)	(**4.94E−02**)
SF4	9.40E+02	2.59E+02	1.09E+03	5.22E+02	4.75E+00	**5.68E−14**
	(7.88E+01)	(1.29E+01)	(6.13E+01)	(2.28E+01)	(2.27E+00)	(**0.00E+ 00**)
SF5	9.42E+02	8.13E+02	1.27E+03	7.92E+02	**6.20E+00**	3.09E+02
	(7.76E+01)	(4.28E+01)	(1.11E+02)	(2.14E+01)	(**5.01E+00**)	(6.23E+01)
EF1	7.45E+01	4.01E+01	1.07E+02	5.46E+01	1.00E+01	**8.16E+00**
	(3.69E+00)	(6.23E−01)	(8.03E+00)	(1.98E+02)	(8.91E−01)	(**1.21E+00**)
EF2	4.74E+01	4.71E+01	4.76E+01	4.73E+01	4.89E+01	**4.58E+00**
	(1.82E−01)	(2.44E−01)	(1.97E−01)	(2.54E−01)	(2.41E−01)	(**6.56E−01**)

Table 4
p-values of Wilcoxon test for $D = 25$ and $D = 100$.

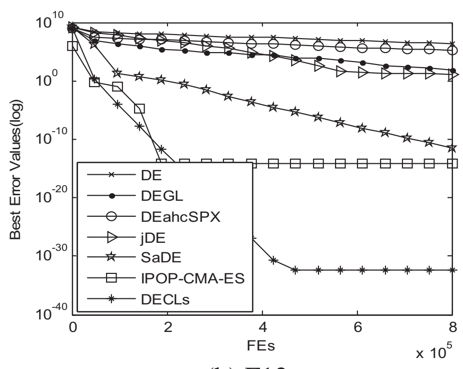
DECLS vs.	$D = 25$	$D = 100$
DEGL	9.6733E−04	8.8575E−05
jDE	4.8625E−02	8.8575E−05
DEahcSPX	2.2252E−03	1.3183E−04
SaDE	6.2103E−03	1.3183E−04
IPOP-CMA-ES	3.7589E−01	2.7724E−01

4.3.4. Convergence performance

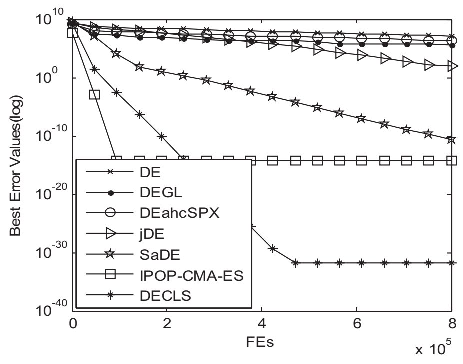
Convergence speed is very important for the evolutionary optimization [5,9]. Many algorithms have good convergence performance on simple functions but they slow down with an increase in function complexity. To show the convergence performance of the proposed DECLS algorithm, we compared it with the DE variants and IPOP-CMA-ES over seven benchmark functions each with different characteristics. We chose one unimodal function (F2), two multimodal function (F12,



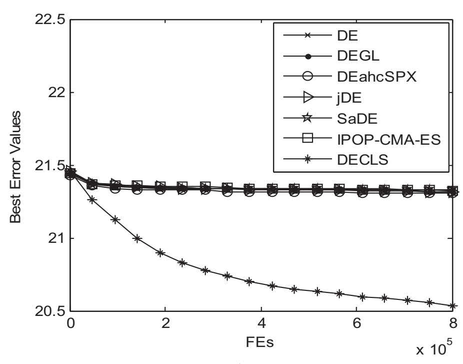
(a) F2



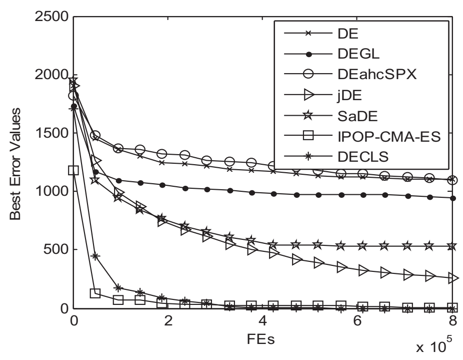
(b) F12



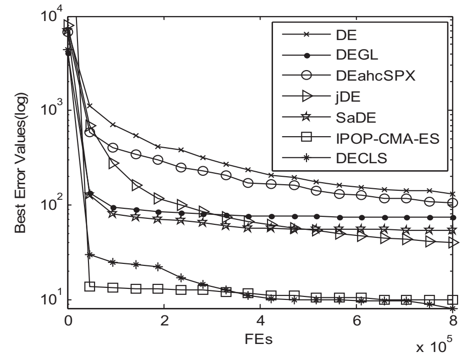
(c) F13



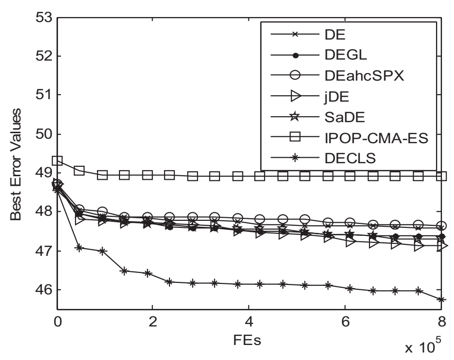
(d) SF3



(e) SF4



(f) EF1



(g) EF2

Fig. 2. Convergence performance of the algorithms for comparison.

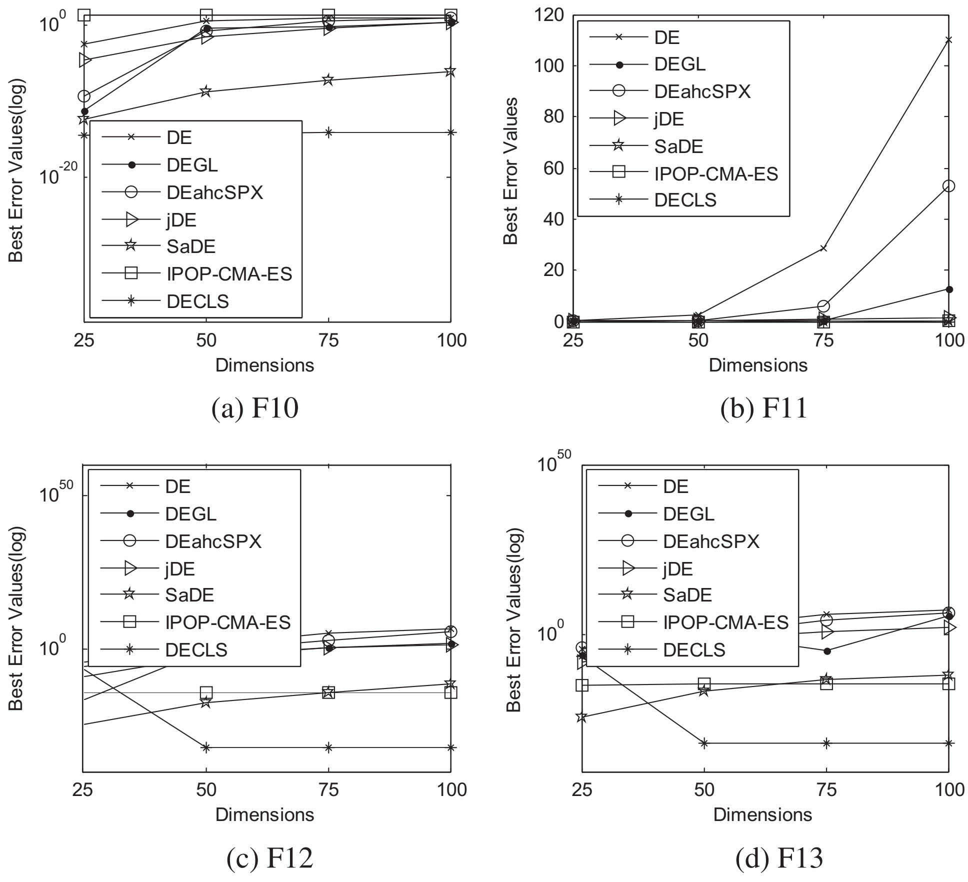


Fig. 3. Scalability comparison over four test problems.

Table 5
Results of DECLS with different *m* values.

m	F4	F5	F8
500	2.49E−03 ± 6.71E−03	2.12E−02 ± 6.34E−02	5.26E+01 ± 6.24E+01
1000	2.07E−03 ± 5.85E−03	1.17E−01 ± 1.87E−01	9.21E+01 ± 7.89E+01
1500	9.93E−05 ± 2.05E−04	8.86E−01 ± 1.76E+00	7.89E+01 ± 8.37E+01
2000	4.18E−05 ± 1.21E−04	9.90E−01 ± 2.97E−01	5.26E+01 ± 6.24E+01
2500	2.38E−05 ± 4.29E−05	1.14E+00 ± 1.32E+00	3.95E+01 ± 5.92E+01

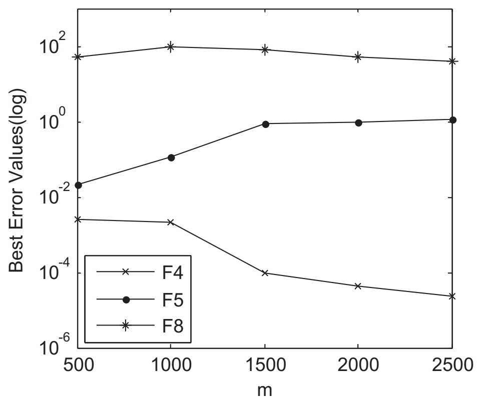


Fig. 4. Results of DECLS with different *m* values.

F13), two shift function (SF3, SF4), and two expand Function (EF1, EF2) at dimension $D = 100$ as the test problems. The convergence graphs of the algorithms over test problems are shown in Fig. 2. Please note that the convergence graphs are the mean best function error values of 25 independent runs at each function evaluation. From the Fig. 2, it can be seen that the DECLS algorithm not only has the fastest convergence speed (the convergence graph of the DECLS is always lower than those of the others), but it obtains the best solutions to the seven test functions.

4.3.5. Scalability

The complexities of the functions increase rapidly with the growth of the dimensionality of the search space which in turn results in a slow convergence of some optimizers. To save space, the scalability comparison shown in Fig. 3 is executed only on four selected test problems with dimensionality from 25 to 100. But please note that the omitted results follow a similar trend as these four functions. From Fig. 3, it can be seen that the performance of some DE variants deteriorates with the growth of the dimensionality of the search space. However, DECLS shows a good scalability by not sensitive to function dimensions.

4.4. Study of the parameter m of DECLS

In all experiments, we fixed $m = 1500$ as the parameter setting for DECLS. However, as mentioned earlier, the parameter m controls the shrinking speed of the search space which further affects the final solution accuracy. Effective choice of m depend on the problem being solved. To illustrate this point, we run DECLS algorithm with different m values over three typical functions. The results of 25 independent runs are summarized in Table 5 and illustrated in Fig. 4.

Table 5 and Fig. 4 show that different problems have different responses to different $m$values. When considering all benchmark functions in our experiments, we choose a fixed value $m = 1500$ for the comparisons.

5. Conclusions

DE is a recently developed simple yet powerful evolutionary algorithm. Due to ease of implementation, DE has been applied to many real-life problems. However, DE does not guarantee the convergence to the global optimum. It is easily trapped into local optima resulting in a low optimizing precision or even a failure. To enhance the optimizing performance of DE, we proposed a memetic evolution algorithm by combining a DE with a CLS and a parameter adaptation mechanism in this paper. By nonlinearly shrinking the search space of CLS, CLS can avoid premature convergence by searching in a huge space in the early run phase and refine the solutions in a small region in the later run phase. The parameter adaptation mechanism further enhances the optimization performance of DE to some extent.

We have demonstrated that DECLS is a superior optimization method for different classes of problems. Results show that the DECLS is better than or at least comparable to four other state-of-the-art DE variants and the IPOP-CMA-ES algorithm in terms of the convergence performance and final solution accuracy. Additionally, results show that DECLS is a promising method for optimizing high dimensional problems.

In the experiments, we also found that the parameter m of CLS is very problem dependent. In our future study, we will provide some empirical or theoretical guidelines for adaptively controlling it to adapt to different types of optimization problems. In addition, we will apply the proposed algorithm to solve some real-world problems in the future.

Acknowledgements

This paper is partially supported by the National Nature Science Foundation of China (Grant No. 60872021) and Shanghai Leading Academic Discipline Project, STCSM (S30108 and 08DZ2231100).

The authors thank Dr. WeiDong Xiang and Chuck Molnar at the Electrical and Computer Engineering Department, University of Michigan-Dearborn, for their help to improve the linguistic quality of this paper.

Finally, we are very grateful to the anonymous associate editor and the anonymous referees for their constructive comments and helpful suggestions to improve the quality of this paper.

Appendix A. Benchmark functions

(1) $F1(X) = \sum_{i=1}^{D} x_i^2;\ -100 \leqslant x_i \leqslant 100;\ F1^*(X) = F1(0,\ldots,0) = 0.$

(2) $F2(X) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} x_i;\ -10 \leqslant x_i \leqslant 10;\ F2^*(X) = F2(0,\ldots,0) = 0.$

(3) $F3(X) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2;\ -100 \leqslant x_i \leqslant 100;\ F3^*(X) = F3(0,\ldots,0) = 0.$

(4) $F4(X) = \max|x_i|;\ -100 \leqslant x_i \leqslant 100;\ F4^*(X) = F4(0,\ldots,0) = 0.$

(5) $F5(X) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right];\ -30 \leqslant x_i \leqslant 30;\ F5^*(X) = F5(1,\ldots,1) = 0.$

(6) $F6(X) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2;\ -100 \leqslant x_i \leqslant 100;\ F6^*(X) = F6(x_1,\ldots,x_D) = 0;\ -\frac{1}{2} \leqslant x_i < \frac{1}{2}.$

(7) $F7(X) = \sum_{i=1}^{D} i x_i^4 + rand[0,1);\ -1.28 \leqslant x_i \leqslant 1.28;\ F7^*(X) = F7(0,\ldots,0) = 0.$

(8) $F8(X) = \sum_{i=1}^{D} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) + D \cdot 418.98288727243369;\ -500 \leqslant x_i \leqslant 500;\ F8^*(X) = F8(0,\ldots,0) = 0.$

(9) $F9(X) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right];\ -5.12 \leqslant x_i \leqslant 5.12;\ F9^*(X) = F9(0,\ldots,0) = 0.$

(10) $F10(X) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e;\ -32 \leqslant x_i \leqslant 32;\ F10^*(X) = F10(0,\ldots,0) = 0.$

(11) $F11(X) = \frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1;\ -600 \leqslant x_i \leqslant 600;\ F11^*(X) = F11(0,\ldots,0) = 0.$

(12) $F12(X) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D}u(x_i, 10, 100, 4);\ -50 \leqslant x_i \leqslant 50;$ $F12^*(X) = F12(-1,\ldots,-1) = 0.$
where,

$$y_i = 1 + \frac{1}{4}(x_i + 1) \quad \text{and} \quad u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a, \\ 0 - a \leqslant & x_i \leqslant a, \\ k(-x_i - a)^m & x_i < -a. \end{cases}$$

(13) $F13(X) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})]\right\} + (x_D - 1)^2\{1 + \sin^2(2\pi x_n)\} + \sum_{i=1}^{D}u(x_i, 5, 100, 4);\ -50 \leqslant$ $x_i \leqslant 50;\ F13^*(X) = F13(1,\ldots,1,1) = 0.$

(14) SF1: Shifted Rosenbrock's Function, $-100 \leqslant x_i \leqslant 100$; $SF1^*(X) = 390$.

(15) SF2: Shifted Rotated Griewank's Function without Bounds, $0 \leqslant x_i \leqslant 600$; $SF2^*(X) = -180$.

(16) SF3: Shifted Rotated Ackley's Function with Global Optimum on Bounds, $-32 \leqslant x_i \leqslant 32$; $SF3^*(X) = -140$.

(17) SF4: Shifted Rastrigin's Function, $-5 \leqslant x_i \leqslant 5$; $SF4^*(X) = -330$.

(18) SF5: Shifted Rotated Rastrigin's Function, $-5 \leqslant x_i \leqslant 5$; $SF5^*(X) = -330$.

(19) EF1: Shifted Expanded Griewank's plus Rosenbrock's Function (F8F2), $-3 \leqslant x_i \leqslant 1$; $EF1^*(X) = -130$.

(20) EF2: Shifted Rotated Expanded Scaffer's F6 Function, $-100 \leqslant x_i \leqslant 100$; $EF2^*(X) = -300$.

References

[1] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2005, pp. 1769–1776.
[2] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Selfadapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation 10 (6) (2006) 646–657.
[3] J. Brest, A. Zamuda, B. Boskovic, M.S. Maucec, V. Zumer, High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, in: 2008 IEEE World Congress on Computational Intelligence, 2008, pp. 2032–2039.
[4] S. Das, P.N. Suganthan, Differential evolution – a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[5] S. Das, A. Konar, U.K. Chakraborty, Two improved differential evolution schemes for faster global search, in: H. Beyer (Ed.), Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO'05, 2005, pp. 991–998.
[6] S. Dasgupta, A. Biswas, S. Das, A. Abraham, Modeling and analysis of the population dynamics of differential evolution algorithm, AI Communications – The European Journal on Artificial Intelligence 221 (1) (2009) 1–20.
[7] S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm, Information Sciences 180 (8) (2010) 1237–1256.
[8] 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) 526–552.
[9] S. Ghosh, S. Das, S. Das, On the Asymptotic Convergence of Differential Evolution in Continuous Spaces – A Control Theoretic Approach, Genetic and Evolutionary Computing Conference (GECCO), Portland, Oregon, USA, 2010, pp. 2073–2074.
[10] S. Garcia, D. Molina, M. Lozano, 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 15 (2009) 617–644.
[11] R. Joshi, A.C. Sanderson, Minimal representation multisensory fusion using differential evolution, IEEE Transaction on Systems, Man and Cybernetics, Part A 29 (1) (1999) 63–76.
[12] D.L. Jia, Y.M. Jiao, J.D. Zhang, Satisfactory design of IIR digital filter based on chaotic mutation particle swarm optimization, in: Third International Conference on Genetic and Evolutionary Computing, 2009, pp. 48–51.
[13] D.L. Jia, J.S. Zhang, Niche particle swarm optimization combined with chaotic mutation, Control and Decision 22 (1) (2007) 117–120.
[14] T. Kapitaniak, Continuous control and synchronization in chaotic systems, Chaos, Solitons and Fractals (6) (1995) 237–244.
[15] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing, 2000, pp. 76–83.
[16] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing: A Fusion Foundations, Methodologies and Applications 9 (6) (2005) 448–462.
[17] G.C. Liao, T.P. Tsao, Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting, IEEE Transaction on Evolutionary Computation 10 (3) (2006) 330–340.
[18] Q.Z. Lü, G.L. Shen, R.Q. Yu, A chaotic approach to maintain the population diversity of genetic algorithm in network training, Computational Biology and Chemistry 27 (3) (2003) 363–371.
[19] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of Genetic Evolutionary Computation Conference, Seattle, 2006, pp. 485–492.
[20] S. Muelas, A. LaTorre, J.M. Pena, A memetic differential evolution algorithm for continuous optimization, in: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, IEEE Computer Society, 2009, pp. 1080–1084.
[21] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 107–125.
[22] G.C. Onwubolu, Design of hybrid differential evolution and group method of data handling networks for modeling and prediction, Information Sciences 178 (18) (2008) 3616–3634.
[23] K. Price, R. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, first ed., Springer-Verlag, New York, 2005.
[24] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1785–1791.

[25] 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.
[26] R. Storn, K. Price, Minimizing the real functions of the ICEC'96 contest by differential evolution, in: Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 842–844.
[27] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Nanyang Technol. Univ., Singapore, Tech. Report and IIT, Kanpur, India, KanGAL Report #2005005, 2005.
[28] Y.W. Shang, Y.H. Qiu, A note on the extended Rosenbrock function, Evolutionary Computation 14 (1) (2006) 119–126.
[29] M.S. Tavazoei, M. Haeri, Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms, Applied Mathematics and Computation 187 (2) (2007) 1076–1085.
[30] Y. Wang, B. Li, T. Weise, Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems, Information Sciences 180 (12) (2010) 2405–2420.
[31] L. Wang, D.Z. Zheng, Lin QS, Survey on chaotic optimization methods, Computing Technology and Automation 20 (1) (2001) 1–5.
[32] H. Wang, Z.J. Wu, S. Rahnamayan, L. Kang, A scalability test for accelerated DE using generalized opposition-based learning, in: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, IEEE Computer Society, 2009, pp. 1090–1095.
[33] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation. 3 (2) (1999) 82–102.
[34] J. Zhang, V. Avasarala, R. Subbu, Evolutionary optimization of transition probability matrices for credit decision-making, European Journal of Operational Research 20 (2) (2010) 557–567.
[35] M. Zhang, W. Luo, X.F. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178 (15) (2008) 3043–3074.
[36] J.Q. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.