Differential Evolution with Multi-Stage Strategies for Global Optimization

Xiao-Gen Zhou, Gui-Jun Zhang*, Xiao-Hu Hao, Li Yu and Dong-Wei Xu,
College of Information Engineering
Zhejiang University of Technology
Hangzhou, Zhejiang, PR China
Email: zgj@zjut.edu.cn

***Abstract*—Differential evolution is a fast, robust, and simple population-based stochastic search algorithm for global optimization, which has been widely applied in various fields. However, there are many mutation strategies in DE, which have their own characteristics. Therefore, choosing a best mutation strategy is not easy for a specific problem. Different mutation strategies may be appropriate during different stages of the evolution. In this paper, we propose a DE with multi-stage strategies (DEMS). In DEMS, the evolution process of DE is divided into multiple stages according to the average distance between each individual in the initial population. Each stage has its own strategy candidate pool which includes multiple effective strategies. At the beginning of each generation, the average distance between each individual is first calculated to determine the evolution stage. Then for each target vector in the current population, a mutation strategy is randomly selected from the strategy candidate pool with respect to the stage to produce a offspring vector. Numerical experiments on 15 well-known benchmark functions and the CEC 2015 benchmark sets show that the proposed DEMS is significantly better than, or at least comparable to several state-of-the-art DE variants, in terms of the quality of the final solutions and the convergence rate.**

I. Introduction

Evolutionary algorithms (EAs), inspired by the natural evolution of species, have been successfully applied to numerous optimization problems in various fields [1], [2], [3]. EAs start by initializing a population of individuals. At each iteration, a mutation or recombination process is used to generate offspring individuals. According to the quality of the offspring individual evaluated by a fitness function, a selection process is applied to select better individuals to update the population.

Differential evolution (DE), as one of the most powerful EAs for global optimization over the continuous search space, is proposed by Storn and Price [4]. DE guides the optimization search process according to the swarm intelligence generated by the cooperation and competition of individuals in the population. DE has been proven to have many advantages, such as ease of use, speed, simple structure, and robustness. These advantages make DE attractive to applications of diverse real-world optimization problems [5], such as communication, power systems [6], chemical [7], bioinformatics [8], and engineering design [9].

In DE, there are many mutation strategies, and they have different characteristics. Some strategies may be good at exploring the search space and locating the region of global optimum, but they are slow at exploitation of the solutions, while other strategies are the opposite. Therefore, it is difficult to choose a suitable mutation strategy for a specific problem. An unsuitable mutation strategy may lead to high computational costs and premature convergence. To select a suitable strategy for a specific problem at hand, researchers have proposed many methods to adaptively choose the mutation strategy for different stages of the search process. Qin *et al.* [10] proposed a DE with strategy adaptation (SaDE), in which the mutation strategy is selected from the strategy candidate pool according to the previous experience. Gong *et al.* [11] proposed a strategy adaptation mechanism (SaM). In SaM, a strategy parameter η is used to control the selection of different strategies, while the parameter η is adaptively updated. Wang *et al.* [12] proposed a composite DE (CoDE). In CoDE, the offspring individual is generated by randomly combining the mutation strategy from the strategy candidate pool with the control parameter from the parameter candidate pool. Mallipeddi *et al.* [13] proposed a DE with ensemble of parameters and mutation strategies (EPSDE). EPSDE includes a pool of distinct mutation strategies and a pool of different control parameters. Each individual selected a mutation strategy and parameters randomly from respective pools to generate the offspring individual.

In this paper, we propose a DE with multi-stage strategies (DEMS). In DEMS, the whole search process is divided into three stages according to the average distance between each individual in the current population. Each stage has its own strategy candidate pool which includes multiple mutation strategies. The mutation strategies in the same pool has the similar characteristics, but different from the strategies in the other pool. At each iteration of each stage, a mutation strategy is randomly selected from the strategy candidate pool that corresponded to the stage to generate the offspring individual. In order to evaluate the performance of the proposed DEMS, four state-of-the-art DE variants are compared with DEMS on 15 commonly used benchmark functions and the CEC 2015 shifted and rotated benchmark sets. Experimental results indicate that our proposed DEMS is able to balance the exploration and the exploitation of the algorithm, thus obtains the faster convergence speed and better quality solutions for a specific problem.

The reminder of this paper is organized as follows. The basic DE and some of its variants are briefly reviewed in Section II. Section III describes the proposed DEMS in detail. Experimental results demonstrating the performance of the proposed DEMS in comparison with four state-of-the-art DE variants over a suite of bound-constrained benchmark functions and the CEC 2015 benchmark sets are presented in Section IV. Section V concludes this paper.

II. Differential evolution and its variants

A. Differential evolution

Differential evolution [4] is a population-based stochastic search algorithm for global optimization. Similar to other EAs, DE also includes initialisation, mutation, crossover and selection operation. Overall, DE is a simple evolutionary algorithm and often performs better than the traditional EAs because of the greedy selection strategy. The details of DE are described as follows.

Suppose that the objective function to be minimized is $f(x), x = (x_1, x_2, \cdots, x_D) \in R^D$, where D is the dimension of $f(x)$. Firstly, at generation $g = 0$, an initial population $P = \{x_i^0 = (x_{i,1}^0, x_{i,2}^0, \cdots, x_{i,D}^0),\ i = 1, 2, \cdots, NP\}$ is randomly sampled from the feasible solution space, where NP is the population size. Then the mutation and crossover operation are used to generate the trial individuals. Finally, the selection operation is applied to determine whether the trial vector will survive to the next generation or not.

1) Mutation Operation: To drive the population to explore the search space, the mutation operation is adopted. At each generation g, DE creates a mutation vector $v_i^g = (v_{i,1}^g, v_{i,2}^g, \cdots, v_{i,D}^g)$ for each individual x_i^g (called a target vector) in the current population. Some well-known mutation strategies are listed as follows:

- "DE/rand/1":
$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{1}$$
- "DE/rand/2":
$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) + F \cdot (x_{r_4}^g - x_{r_5}^g) \tag{2}$$
- "DE/best/1":
$$v_i^g = x_{\text{best}}^g + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{3}$$
- "DE/best/2":
$$v_i^g = x_{\text{best}}^g + F \cdot (x_{r_1}^g - x_{r_2}^g) + F \cdot (x_{r_3}^g - x_{r_4}^g) \tag{4}$$
- "DE/current-to-best/1":
$$v_i^g = x_i^g + F \cdot (x_{\text{best}}^g - x_i^g) + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{5}$$
- "DE/rand-to-best/1":
$$v_i^g = x_{r_1}^g + F \cdot (x_{\text{best}}^g - x_{r_1}^g) + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{6}$$
- "DE/current-to-rand/1":
$$v_i^g = x_i^g + F \cdot (x_{r1}^g - x_i^g) + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{7}$$

where r_1, r_2, r_3, r_4, and r_5 are distinct integers randomly selected from the range $[1, NP]$ and are also different from i. The parameter F is called the scaling factor, which amplifies the difference variation. x_{best}^g is the best vector in aspects of function value at the current generation g.

2) Crossover Operation: To enhance the potential diversity of the population, a crossover operation is performed after mutation. DE performs a binomial crossover operation on x_i^g and v_i^g to generate a trial vector $u_i^g = (u_{i,1}^g, u_{i,2}^g, \cdots, u_{i,D}^g)$. The trial vector is generated as

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

where $j = 1, 2, \cdots, D$, j_{rand} is a randomly chosen index from $[1, D]$, $\text{rand}_j(0,1)$ is a uniformly distributed random number generated from $[0,1]$ for each j, and $CR \in (0,1)$ is called the crossover control parameter. Due to the use of j_{rand}, the trial vector u_i^g differs from its target vector x_i^g.

3) Selection Operation: The purpose of selection operation is to select the better one from the target vector x_i^g and the trial vector u_i^g to enter the next generation. The selection operation is described as

$$x_i^{g+1} = \begin{cases} u_i^g, & \text{if } \ f(u_i^g) \le f(x_i^g) \\ x_i^g, & \text{otherwise} \end{cases} \tag{9}$$

Namely, if the trial vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next generation; otherwise the target vector is retained in the population. Hence the population either gets better or remains the same in function status, but never deteriorates.

B. Some variants of DE

In this section, some variants of DE, i.e., SaDE [10], SaM [11], JADE [14], CoDE [12], and EPSDE [13] are described as follows:

1) SaDE: In SaDE [10], the strategy candidate pool includes K mutation strategies. The probability of being selected for the kth strategy is denoted as p_k. p_k is initialized as $1/K$. At each iteration, a strategy is adaptively selected according to the probability p_k. After LP generation, p_k is updated in the following way:

$$p_k = \frac{Sk, g}{\sum_{k=1}^{K} Sk, g} \tag{10}$$

where

$$Sk, g = \frac{\sum_{t=g-LP}^{g-1} ns_{k,t}}{\sum_{t=g-LP}^{g-1} ns_{k,t} + \sum_{t=g-LP}^{g-1} nf_{k,t}} + \epsilon \tag{11}$$

where $g(g > LP)$ represents the number of generation; $ns_{k,g}$ and $nf_{k,g}$ are the number of offspring individuals generated by the kth strategy that survive or fail in the selection operation at generation g, respectively. $S_{k,g}$ represents the success rate of the offspring individuals generated by the kth strategy and survive in the selection operation within the previous LP generations. ϵ is a small constant to avoid the possible null success rates.

2) SaM: Gong *et al.* [11] proposed the SaM method, in which a strategy parameter η_i is adopted to control the selection of the strategy for the ith target vector x_i. Suppose that the strategy candidate pool includes K strategies, then the mutation strategy $S_i = \{1, 2, \cdots, K\}$ for the ith target vector can be obtained as

$$S_i = \lfloor \eta_i \times K \rfloor + 1. \tag{12}$$

For example, it means that the first strategy in the pool will be selected to generate the mutant vector for the ith target vector if $S_i = 1$. Moreover, the parameter η_i can be generated using three approaches. Here, the second approach is described as below, details of the other two approaches can be found in [11]. For the ith target vector, the parameter η_i in the second approach is calculated as

$$\eta_i = \begin{cases} \text{rndreal}[0,1), & \text{if} \quad \text{rndreal}[0,1) < \delta \\ \eta_i, & \text{otherwise} \end{cases} \tag{13}$$

where rndreal[0,1) is a uniformly distributed random number generated in [0,1). $\delta \in [0,1]$ represents the probability to adjust the strategy parameter η_i.

3) ADE: Yu *et al.* [15] proposed a DE with two-level parameter adaption (ADE). In ADE, a new mutation strategy called "DE/lbest/1" is proposed to balance the fast convergence and population diversity. In the proposed strategy, the entire population is randomly divided into several nonoverlapping groups which include the same number of individuals. Then the best vector is selected from the group which the target vector belongs. The "DE/lbest/1" can be expressed as follows:

$$v_i^g = x_{\text{lbest}}^g + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{14}$$

where x_{lbest}^g is the best vector of the group with respect to the target vector.

Moreover, the control parameters in ADE are adaptively selected. Briefly, the optimization state is first estimated according to the rankings of the fitness values and the rankings of the Euclidean distances from the best individual to the other individuals. Then the population-level parameter values F_p and CR_P are adaptively adjusted according to the estimated optimization state. Finally, the individual-level parameter values F_i and CR_i for each individual are generated based on the values of F_p and CR_p.

4) JADE: Zhang *et al.* [14] proposed an adaptive DE with optional external archive, namely, JADE. In JADE, two new mutation strategies called "DE/current-to-*p*best" without archive and "DE/current-to-*p*best" with archive are proposed to improve the performance of DE. They generate mutation vectors in the following manner:

- "DE/current-to-*p*best/1 (without archive)":

$$v_i^g = x_i^g + F \cdot (x_{\text{best},g}^p - x_i^g) + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{15}$$

- "DE/current-to-*p*best/1 (with archive)":

$$v_i^g = x_i^g + F \cdot (x_{\text{best},g}^p - x_i^g) + F \cdot (x_{r_1}^g - \widetilde{x}_{r_2}^g) \tag{16}$$

where $x_{\text{best},g}^p$ is randomly chosen from the top $100p\%$ individuals in the current population with $p \in (0,1]$. $\widetilde{x}_{r_2}^g$ is randomly chosen from the union $\mathbf{P} \cup \mathbf{A}$, where $\mathbf{P}$ and $\mathbf{A}$ represent the current population and the set of archived inferior solutions, respectively.

In JADE, each individual x_i has its own crossover rate CR_i and scaling factor F_i. The CR_i and F_i of each individual at each generation are generated according to

$$CR_i = \text{randn}_i(\mu CR, 0.1) \tag{17}$$

$$F_i = \text{randc}_i(\mu F, 0.1) \tag{18}$$

where $\text{randn}_i(\mu, \sigma^2)$ and $\text{randc}_i(\mu, \sigma^2)$ represent the normal and Cauchy distributions with mean μ and standard deviation σ^2. If CR_i or F_i are outside [0,1], they are truncated to [0,1] or regenerated.

For the initial population, μCR and μF are initialized to 0.5, and updated at the end of each generation as follows.

$$\mu CR = (1 - c) \cdot \mu CR + c \cdot \text{mean}_A(S_{CR}) \tag{19}$$

$$\mu F = (1 - c) \cdot \mu F + c \cdot \text{mean}_L(S_F) \tag{20}$$

where s_{CR} and S_F are the CR_i and F_i that generated trial vector survived in the selection operation, respectively. c is a positive constant between 0 and 1. mean_A is the arithmetic mean, and mean_L is the Lehmer mean which is calculated according to

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{21}$$

III. DEMS ALGORITHM

The performance of DE highly depends on the selected mutation strategy. However, there are many mutation strategies in DE, and they have distinct characteristics. For a specific problem, choosing a suitable mutation strategy is not easy for DE. Different mutation strategies are used at different stages of the search process may achieve the better performance. In addition, multiple candidate strategies for each stage may be better than a single mutation strategy. Motivated by these observations, we proposed a DE with multi-stage strategies (DEMS). In DEMS, the whole evolution process is divided into three stages according to the average distance between each individual in the population. Each stage has its own strategy candidate pool. The strategies in the same candidate pool have the similar characteristics, but different from strategies in the other candidate pool. For each target vector at each stage, a mutation strategy is randomly selected from the strategy candidate pool with respect to the stage to generate a trial vector. In addition, the control parameters are adaptively selected according to the technique proposed in JADE. The DEMS is described in detail below.

A. Stage division

In DE, all individuals are scattered in the initial population. At each generation, all individuals gradually converge to the global optimum. Finally, all individuals may be attracted into several promising regions or converge to an optimum solution. In DEMS, the average distance between each individual in the current population is used to estimate the convergence condition. The average distance between each individual of each generation can be calculated as

$$d_g = \frac{\sum_{i=1}^{NP} \sum_{k=i+1}^{NP} ||x_i^g - x_k^g||}{NP(NP-1)/2} \tag{22}$$

where NP is the population size. In the initial population, the average distance between each individual has the maximum value $d_{\max} = d_0$. For the single global optimization problem, the goal is to make all individuals converge to an optimum solution finally. Thus, the average distance has its minimum value $d_{\min} = 0$.

According to the average distance d_g of each generation, we can estimate the evolution stages based on

$$\Phi = \begin{cases} S_1, & \text{if } \ d_g > 2sd_{\max} \\ S_2, & \text{if } \ sd_{\max} < d_g \leq 2sd_{\max} \\ S_3, & \text{otherwise} \end{cases} \tag{23}$$

where Φ is the optimization stage, S_1, S_2, and S_3 represent the first, second, and third stage of the optimization, respectively, and $s \leq 1/3$ is the stage control parameter.

B. Selection of mutation strategies for each stage

As mentioned above, there are many mutation strategies in DE, and different strategy has different characteristics. In the following, nine frequently used mutation strategies are investigated and they are used in this paper.

- The "DE/rand/1/bin" and "DE/rand/2/bin" strategy are two classic mutation strategies, which have been widely used in many DE literatures. It has been proven that they has slower convergence speed and stronger exploration capabilities. Hence, they are more suitable for solving multimodal problems.
- "DE/current-to-rand/1" is a rotation-invariant strategy without crossover. It is very effective for multiobjective optimization problems [16].
- "DE/current-to-*p*best/1/bin" with archive [14] has been verified can balance the greediness of the mutation and the diversity of the population by using the information of multiple best solutions [14]. Moreover, "DE/rand-to-*p*best/1/bin" with archive proposed by Zhang *et al.* [17] and "DE/*p*best/2/bin" with archive are also adopted in our proposed algorithm.
- "DE/lbest/1/bin" [15] which uses the information of the locally best solutions of the population, is helpful for balancing the exploitation and the population diversity. In addition, "DE/rand-to-lbest/1/bin" and "DE/current-to-lbest/1/bin" are also used in the proposed DEMS.

According to above analysis of characteristic for each mutation strategy, we can select strategies for each stage. For the first stage, the algorithm explores the promising regions which may include the global optimum. Therefore, the "DE/rand/1/bin", "DE/rand/2/bin", and "DE/current-to-rand/1" used in this stage is more suitable. For the second stage, in order to explore more promising regions and exploits the regions that have been found, "DE/current-to-*p*best/1/bin" with archive, "DE/rand-to-*p*best/1/bin" with archive, and "DE/*p*best/2/bin" with archive are adopted in this stage. For the last stage, the algorithm begins to exploits the global optimum. To improve the convergence rate and prevent the algorithm from getting trapped into a local minimum, "DE/lbest/1/bin", "DE/current-to-lbest/1/bin", and "DE/rand-to-lbest/1/bin" are used in this stage. The strategy candidate pool for each stage is listed as follows:

1) S_1 stage strategy pool:
   - "DE/rand/1/bin";
   - "DE/rand/2/bin";
   - "DE/current-to-rand/1".
2) S_2 stage strategy pool:
   - "DE/*p*best/2/bin" with archive;
   - "DE/current-to-*p*best/1/bin" with archive;
   - "DE/rand-to-*p*best/1/bin" with archive.
3) S_3 stage strategy pool:
   - "DE/lbest/1/bin";
   - "DE/current-to-1best/1/bin";
   - "DE/rand-to-1best/1/bin".

For each target vector at each stage, a mutation strategy is randomly selected from the corresponding stage strategy pool to generate a trial vector.

C. Parameter adaption

DE has two main parameters scaling factor (F) and crossover rate (CR) that affect the search process. In this paper, the parameter adaption technique propose in JADE [14] is used in the proposed DEMS. Although the parameters CR and F are adaptively selected in JADE, the parameter c, which is used to control the learning rate of μCR, is fixed and set manually. In our proposed DEMS, each generation has its own c_g which is calculated according to change rate of the average distance between each individual in the population. c_g is initialized to 0.1 and then updated at the end of each generation according to the following manner:

$$c_g = \frac{|d_{g-1} - d_g|}{d_{g-1}} \tag{24}$$

where d_g and d_{g-1} are the average distances between each individual of the current generation g and the last population $g-1$, respectively. If $c_g > 1$, it is truncated to [0,1].

D. Overall implementation

In DEMS, the average distance between each individual in the initial population is firstly calculated according to (22), and it is regarded as the maximum value $d_{\max}$ of the average distance. Based on the $d_{\max}$, the whole search process is divided into three stages which have their own strategy candidate pool according to (23). At the beginning of each generation, the average distance between each individual is calculated to estimate the evolution stage of the generation. Then for each target vector in the current population, a mutation strategy is randomly selected from the stage candidate pool with respect to the stage to generate a trial vector. Algorithm 1 presents the pseudocode of DEMS.

Algorithm 1. The Pseudocode of DEMS

```
1  Randomly initialize the population $P$ at $g = 0$
2  Evaluate the function value for each individual
3  Calculate $d_0$ according to (22)
4  while the stop rule is not staisfied do
5      Calculate $d_g$ according to (22)
6      Determine the stage according to (23)
7      for $i = 1$ to $N_P$ do
8          Randomly select a strategy from stage strategy pool
9          Generate trial vector $u_i$ using the selected strategy
10     end for
11     for $i = 1$ to $N_P$ do
12         Evaluate the trial vector $u_i$
13         if $f(u_i)$ is better than or equal to $f(x_i)$ then
14             Replace $x_i$ with $u_i$
15         end if
16     end for
17 end while
```

E. Runtime Complexity

In DEMS, the whole search process is divided into three stages according to the stage division method. At each iteration of each stage, a strategy is randomly selected from the corresponding strategy candidate pool to generate a trial vector. For the stage division, we need to calculate the average distance between each individual at each generation. The runtime complexity of the stage division is $O(0.5NP^2 \cdot D \cdot G_{\max})$ since we use Euclidean distance, where $G_{\max}$ is the maximum number of generations, D is the dimension of the problem. The runtime complexity of the DE/rand/1, DE/rand/2, and DE/current-to-rand is $O(NP \cdot D)$. For the DE/*p*best/2, DE/current-to-*p*best/1, and DE/rand-to-*p*best/1, the locally best individual in each group needs to be found. The runtime complexities of them also are $O(NP \cdot D)$. For the DE/lbest/2, DE/current-to-lbest/1, and DE/rand-to-lbest/1, we need to sort all the individuals based on their function values. The sorting process requires $O(NP \cdot \log_2 NP)$ runtime since we use the heap sort algorithm. Hence, the runtime complexities of the DE/*p*best/2, DE/current-to-*p*best/1, and DE/rand-to-*p*best/1 are $O(\max(NP \cdot \log_2 NP, NP \cdot D))$. Suppose that each stage is performed in $G_{\max}/3$ generations, the overall runtime complexity of DEMS is $O(0.5NP^2 \cdot D \cdot G_{\max})$.

IV. Experimental studies

In this section, various experiments on optimization benchmark problems are implemented to verify the performance of the proposed DEMS algorithm.

A. Experimental setup

In order to verify the performance of the proposed DEMS, 15 classical benchmark functions and the CEC 2015 shifted and rotated benchmark sets are used in the following experimental studies. In all experiments, DEMS is compared with four state-of-the-art DE variants, i.e., SaDE [10], jDE [18], EPSDE [13], and JADE [14].

For all experiments, the following parameters for DEMS are used unless a change is mentioned:

- Population size: $NP = 100$.
- $\mu CR = 0.5$, $\mu F = 0.5$.
- $p = 0.05$, $s = 0.1$, $c_0 = 0.1$.
- Stop rule: Each algorithm stops when the number of function evaluations (FES) reaches the maximum FES (MaxFES). For the 15 classical benchmark functions, MaxFES$= 2000 \times D$, and for the CEC 2015 benchmark functions, MaxFES$=10000 \times D$, where D is the dimension of the functions.
- Number of runs: Each function for each algorithm is run 30 times respectively.

For SaDE, jDE, EPSDE, and JADE, the parameter settings are kept the same as in their original papers. For fairly comparison, all the four contestant algorithms use the same stopping rule as DEMS.

B. Performance criteria

Two performance criteria are selected to evaluate the performance of the algorithms. These criteria are described as follows:

1) Error: The function error of a solution x is defined as $f(x) - f(x^*)$, where x^* is the global optimum of the function. The best error of each function in each run is recorded when the MaxFES is reached. The average and standard deviation of the best error are calculated for comparison.
2) Convergence graphs: The convergence graphs show the mean convergence characteristics of the best solution over the total runs, in the respective experiments.

C. Evaluate DEMS using classical benchmark functions

In this section, a set of 15 well-known benchmark functions which are used in DE literatures are used in the following experiments. A brief description of these benchmark functions are given in Table I. A more detailed description of these functions can be found in [11], [19], [15].

Functions $f_1 - f_7$ are unimodal. Function f_6 is a step function that has one minimum and is discontinuous. Function f_7 is the Rosenbrock's function, it becomes multimodal when $D > 3$ [20]. Functions $f_8 - f_{15}$ are multimodal functions, which the local minima increases exponentially with the dimension. All the above functions used in our experiments are problems to be minimized.

Table II summarizes the mean function error value and the corresponding standard deviation obtained by SaDE, jDE, EPSDE, JADE, and DEMS for all the classical benchmark functions at $D = 30$, where "Mean Error" indicates the mean value of the best function error of the 30 independent runs and "Std Dev" indicates the corresponding standard deviation. Moreover, in order to study the difference between any two algorithms in a meaningful way, the paired Wilcoxon signed-rank test [21] at 5% significance level is conducted on the

TABLE I: Fifteen benchmark functions used in the experimental studies

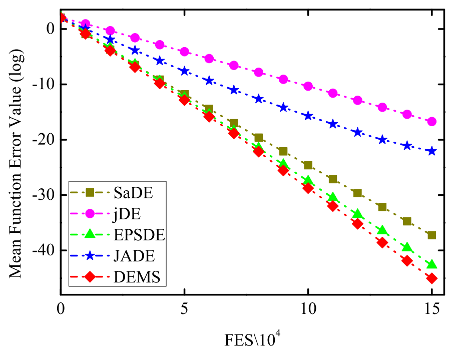
Name	Function	D	Search range	Optimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]$	0
SumSquares	$f_2(x) = \sum_{i=1}^{D} ix_i^2$	30	$[-10, 10]$	0
Schwefel 2.22	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$[-10, 10]$	0
Tablet	$f_4(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	30	$[-100, 100]$	0
Step	$f_5(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	30	$[-100, 100]$	0
Zakharov	$f_6(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4$	30	$[-5, 10]$	0
Rosenbrock	$f_7(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	$[-2, 2]$	0
Griewank	$f_8(x) = 1 + \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})$	30	$[-600, 600]$	0
Schaffer 2	$f_9(x) = \sum_{i=1}^{D-1} (x_i^2 + x_{i+1}^2)^{0.25} (\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$	30	$[-100, 100]$	0
Schwefel 2.26	$f_{10}(x) = -\sum_{i=1}^{D} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	$[-500, 500]$	$-418.983D$
Himmelblau	$f_{11}(x) = \frac{1}{D} \sum_{i=1}^{D} (x_i^4 - 16x_i^2 + 5x_i)$	30	$[-100, 100]$	-78.3323
Ackley	$f_{12}(x) = -20 \exp(-0.2\sqrt{D^{-1} \sum_{i=1}^{D} x_i^2}) - \exp(D^{-1} \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	30	$[-30, 30]$	0
Rastrigin	$f_{13}(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i))$	30	$[-5, 5]^D$	0
Penalized 1	$f_{14}(x) = \frac{\pi}{D} \{\sum_{i=1}^{D-1} (y_i - 1)^2 [1 + \sin(\pi y_{i+1})] + (y_D - 1)^2 + (10\sin^2(\pi y_1))\}$ $+ \sum_{i=1}^{D} u(x_i, 10, 100, 4), y_i = 1 + \frac{x_i + 1}{4}$	30	$[-50, 50]$	0
Penalized 2	$f_{15}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})]$ $+ (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

TABLE II: Comparison of DEMS with SaDE, jDE, EPSDE, and JADE on all benchmark functions at $D = 30$

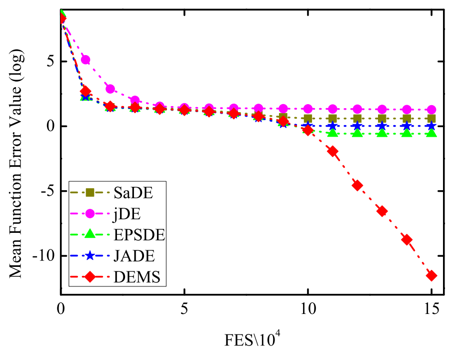
Fun	SaDE	jDE	EPSDE	JADE	DEMS
	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev
f_1	9.00E−68 (1.62E−67)$^{+}$	2.48E−28 (2.95E−28)$^{+}$	2.00E−83 (3.95E−83)$^{+}$	1.76E−57 (4.67E−57)$^{+}$	**2.20E−90 (6.96E−90)**
f_2	9.19E−68 (3.39E−67)$^{+}$	1.00E−29 (8.60E−30)$^{+}$	3.77E−82 (1.42E−81)$^{+}$	1.06E−59 (2.32E−59)$^{+}$	**1.42E−95 (2.63E−95)**
f_3	8.68E−38 (1.02E−37)$^{+}$	1.92E−17 (1.22E−17)$^{+}$	2.13E−43 (4.37E−43)$^{+}$	5.43E−28 (1.48E−27)$^{+}$	**7.77E−47 (2.13E−46)**
f_4	4.43E−67 (1.44E−66)$^{+}$	4.46E−28 (6.27E−28)$^{+}$	7.81E−75 (2.07E−74)$^{+}$	1.65E−54 (6.39E−54)$^{+}$	**2.83E−92 (8.93E−92)**
f_5	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**
f_6	1.31E−13 (1.87E−13)$^{-}$	1.91E−03 (3.01E−03)$^{+}$	1.81E+01 (1.75E+01)$^{+}$	**2.56E−16 (9.84E−16)**$^{-}$	1.51E−11 (3.28E−11)
f_7	2.15E+00 (3.17E+00)$^{+}$	1.89E+01 (9.38E−01)$^{+}$	2.66E−01 (1.03E+00)$^{+}$	1.06E+00 (1.82E+00)$^{+}$	**2.97E−12 (6.96E−12)**
f_8	1.31E−03 (5.08E−03)$^{+}$	**0.00E+00 (0.00E+00)**$^{\approx}$	1.15E−03 (3.07E−03)$^{+}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**
f_9	3.67E−14 (2.91E−14)$^{+}$	1.40E−05 (8.09E−06)$^{+}$	2.23E−08 (4.26E−08)$^{+}$	9.83E−02 (3.22E−02)$^{+}$	**2.36E−15 (2.53E−15)**
f_{10}	**0.00E+00 (0.00E+00)**$^{\approx}$	6.62E−03 (0.00E+00)$^{+}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**
f_{11}	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**$^{\approx}$	**0.00E+00 (0.00E+00)**
f_{12}	**3.55E−15 (0.00E+00)**$^{\approx}$	7.34E−15 (9.17E−16)$^{+}$	4.50E−15 (1.63E−15)$^{+}$	**3.55E−15 (0.00E+00)**$^{\approx}$	**3.55E−15 (0.00E+00)**
f_{13}	4.97E−02 (2.22E−01)$^{+}$	1.18E−15 (4.59E−15)$^{-}$	**8.88E−17 (3.97E−16)**$^{-}$	4.37E−11 (2.29E−11)$^{\approx}$	2.39E−11 (2.11E−11)
f_{14}	**1.57E−32 (0.00E+00)**$^{\approx}$	1.24E−30 (1.28E−30)$^{+}$	**1.57E−32 (0.00E+00)**$^{\approx}$	**1.57E−32 (0.00E+00)**$^{\approx}$	**1.57E−32 (0.00E+00)**
f_{15}	1.46E−03 (3.87E−03)$^{+}$	1.09E−29 (7.18E−30)$^{+}$	1.37E−32 (0.00E+00)$^{\approx}$	**1.35E−32 (0.00E+00)**$^{\approx}$	**1.35E−32 (0.00E+00)**
$+$	9	11	9	6	-
$\approx$	5	3	5	8	-
$-$	1	1	1	1	-

experimental results. We mark with "+" when the first algorithm is significantly better than the second, with "≈" when there is no significant difference between the two algorithms and with "−" when the first algorithm is significantly worse than the second. According to the results summarized in the last three row of the table, we can find that the proposed DEMS outperforms the other four algorithms on the majority of benchmark functions. To be specific, DEMS is significantly better than SaDE on 9 out of 15 functions, including 5 unimodal functions ($f_1 - f_4$ and f_7) and 4 multimodal functions (f_8, f_9, f_{13}, and f_{15}), while SaDE exhibits significantly better performance than DEMS only on one function (f_6). For the other 5 functions, there is no significantly difference between DEMS and SaDE. Compared to jDE, DEMS shows significantly better performance on 11 functions. jDE performs significantly better than DEMS only on one function (f_{13}). EPSDE is significantly superior to DEMS on one function (f_{13}), while DEMS achieves significantly better result than EPSDE on 9 functions. DEMS exhibits significantly better performance than JADE on 6 functions, which JADE significantly outperforms DEMS only on one function (f_6).

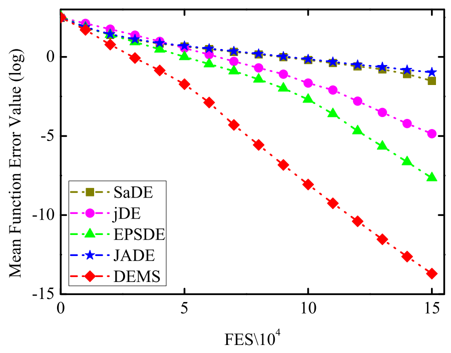
Fig. 1 shows the mean convergence graphs of SaDE, jDE,



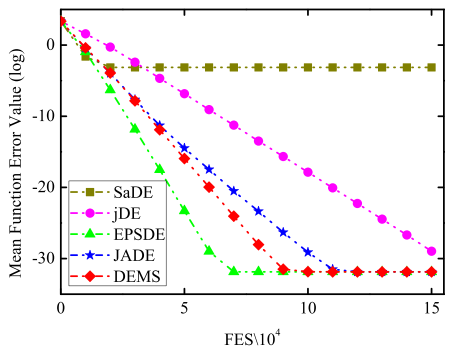
(a) f_3: Schwefel 2.22



(b) f_7: Rosenbrock



(c) f_9: Schaffer 2



(d) f_{15} :Penalized 2

Fig. 1: Mean convergence graphs of SaDE, jDE, EPSDE, JADE, and DEMS on f_3, f_7, f_9, and f_{15}.

TABLE III: Comparison of DEMS with SaDE, jDE, EPSDE, and JADE on CEC 2015 benchmark sets at $D = 30$

Fun	SaDE	jDE	EPSDE	JADE	DEMS
	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev	Mean Error± Std Dev
F_1	1.78E+03 (1.43E+03)$^+$	3.70E+04 (2.12E+04)$^+$	1.71E+06 (2.00E+06)$^+$	6.23E+00 (1.55E+01)$^+$	**6.08E−01 (6.80E−01)**
F_2	2.38E−11 (7.22E−11)$^+$	3.40E−09 (5.64E−09)$^+$	1.82E−12 (3.33E−12)$^+$	**3.41E−14 (1.17E−14)**$^\approx$	3.41E−14 (1.27E−14)
F_3	2.05E+01 (5.99E−02)$^+$	2.03E+01 (2.93E−02)$^\approx$	2.04E+01 (3.00E−02)$^+$	2.03E+01 (2.86E−02)$^\approx$	**2.03E+01 (2.11E−02)**
F_4	3.46E+01 (6.44E+00)$^+$	4.19E+01 (4.03E+00)$^+$	5.25E+01 (7.54E+00)$^+$	2.61E+01 (3.39E+00)$^+$	**2.59E+01 (4.96E+00)**
F_5	3.08E+03 (4.45E+02)$^+$	2.59E+03 (4.03E+02)$^+$	2.71E+03 (2.75E+02)$^+$	**1.70E+03 (2.30E+02)**$^-$	2.17E+03 (2.00E+02)
F_6	3.27E+03 (6.55E+03)$^+$	2.73E+03 (4.82E+03)$^+$	3.96E+05 (5.57E+05)$^+$	**9.59E+02 (3.78E+02)**$^-$	1.10E+03 (4.59E+02)
F_7	**4.97E+00 (2.40E+00)**$^-$	8.08E+00 (7.61E−01)$^+$	9.34E+00 (1.28E+00)$^+$	7.90E+00 (9.56E−01)$^+$	7.73E+00 (1.08E+00)
F_8	6.92E+02 (5.06E+02)$^+$	1.43E+02 (1.00E+02)$^-$	1.79E+03 (2.60E+03)$^+$	4.14E+03 (1.73E+04)$^+$	**2.47E+02 (1.11E+02)**
F_9	1.03E+02 (1.98E−01)$^\approx$	1.03E+02 (1.32E−01)$^\approx$	1.03E+02 (1.69E−01)$^\approx$	1.03E+02 (2.17E−01)$^\approx$	**1.03E+02 (1.19E−01)**
F_{10}	9.15E+02 (4.22E+02)$^+$	7.71E+02 (2.44E+02)$^+$	1.29E+03 (1.15E+02)$^+$	1.93E+04 (5.77E+04)$^+$	**7.08E+02 (1.63E+02)**
F_{11}	4.44E+02 (1.03E+02)$^+$	4.02E+02 (6.07E+01)$^-$	**3.52E+02 (1.60E+01)**$^-$	4.17E+02 (6.74E+01)$^+$	4.04E+02 (7.66E+01)
F_{12}	1.05E+02 (6.95E−01)$^\approx$	1.06E+02 (3.92E−01)$^+$	1.06E+02 (5.14E−01)$^+$	1.05E+02 (3.83E−01)$^\approx$	**1.05E+02 (3.02E−01)**
F_{13}	1.08E+02 (3.40E+00)$^+$	9.98E+01 (2.05E+00)$^+$	1.02E+02 (2.68E+00)$^+$	**9.56E+01 (3.21E+00)**$^-$	9.57E+01 (3.74E+00)
F_{14}	3.31E+04 (1.31E+03)$^+$	3.27E+04 (1.01E+03)$^+$	**3.14E+04 (7.28E+02)**$^-$	3.24E+04 (9.19E+02)$^-$	3.25E+04 (1.12E+03)
F_{15}	**1.00E+02 (0.00E+00)**$^\approx$	**1.00E+02 (0.00E+00)**$^\approx$	**1.00E+02 (0.00E+00)**$^\approx$	**1.00E+02 (0.00E+00)**$^\approx$	**1.00E+02 (0.00E+00)**
$+$	11	10	11	6	-
$\approx$	3	3	2	5	-
$-$	1	2	2	4	-

EPSDE, JADE, and DEMS on some selected functions, i.e., f_3, f_7, f_9, and f_{15}. It is clear that, DEMS always converges faster than the other four algorithms on functions f_3 and f_9. For function f_7, it is the Rosenbrock problem whose global optimum locates in along narrow parabolic shaped flat valley. This makes the algorithm easy to locate the local minimum but difficult to find the global minimum. However, DEMS can converge to the global minimum, while the other four algorithms fall into a local minimum. For function f_{15}, EPSDE, DEMS, and JADE find the global minimum. EPSDE has the fastest convergence, and followed by DEMS. SaDE gets trapped into a local minimum thus lead to premature convergence.

D. Evaluate DEMS using CEC 2015 benchmark sets

In order to further verify the performance of the proposed DEMS, the CEC 2015 shifted and rotated benchmark sets are used in this section. The definitions and properties of them can be found in [22].

Table II presents the results of the mean function error value and the corresponding standard deviation. Functions $F_1 - F_2$ are unimodal function. Functions $F_3 - F_5$ are simple multimodal function. Functions $F_6 - F_8$ are hybrid function. Functions $F_9 - F_{15}$ are composition function. According to the Wilcoxon signed-rank test, the results listed in last three row of the table. From the results, we can find that the proposed DEMS is significantly superior to the other four algorithms on the majority of benchmark functions. In more detail, DEMS significantly outperforms SaDE on 11 out of 15 functions, while SaDE is significantly better than DEMS only on one function (F_7). Compared to jDE, DEMS shows significantly better performance on 10 functions, while jDE performs significantly better than DEMS on 2 function (F_8 and F_{11}). EPSDE achieves significantly better results than DEMS on 2 functions (F_{11} and F_{14}), but DEMS significantly outperforms EPSDE on 11 functions. DEMS is significantly better than JADE on 6 functions, while JADE is significantly superior to DEMS on 4 functions (F_5, F_6, F_{13} and F_{14}).

V. Conclusion

In this paper, we proposed an improved DE variants (DEM-S), in which the mutation strategy is changed along with the evolution stage. In the proposed DEMS, the evolution process are divided into multiple stages according to the average distance between each individual for each generation. Each stage has its own strategy candidate pool and the strategies in the same pool have the similar characteristics. For each target vector, a strategy is randomly selected from the strategy candidate pool with respect to the stage to generate a trial vector. Experiments have been conducted on 15 classical benchmark functions and the CEC 2015 shifted and rotated benchmark sets. The results verify the proposed algorithm can balance the exploration and the exploitation. Compared with some state-of-the art DE variants, the proposed DEMS achieves better results and faster convergence rate on the majority of benchmark functions.

Acknowledgment

This work was supported by National Nature Science Foundation of China (No. 61075062, 61573317), Natural Science Foundation of Zhejiang Province (No. LY13F030008), Open Fund for the Key-Key Discipline of Zhejiang Province (No. 20151008, 20151015). The authors would like to thank Prof. P. N. Suganthan for providing the source code of SaDE, JADE, jDE, and EPSDE, and gratefully acknowledge the anonymous reviewers for their insight comments and useful suggestions.

References

[1] B. Yuan, B. Li, H. Chen, and X. Yao, "A new evolutionary algorithm with structure mutation for the maximum balanced biclique problem," *IEEE Tran. Cybern.*, vol. 45, no. 5, pp. 1040–1053, 2015.

[2] Z. Ding, J. Liu, and C. J. adn M. Zhou, "A transaction and qos-aware service selection approach based on genetic algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 45, no. 7, pp. 1035–1046, 2015.

[3] K. B. Lee and J. H. Kim, "Multiobjective particle swarm optimization with preference-based sort and its application to path following footstep optimization for humanoid robots," *IEEE Tran. Evol. Comput.*, vol. 17, no. 6, pp. 755–766, 2013.

[4] 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.

[5] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution-an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[6] A. Glotic, P. Kitak, J. Pihler, and I. Ticar, "Parallel self-adaptive differential evolution algorithm for solving short-term hydro scheduling problem," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2347–2358, 2014.

[7] S. Sharma and G. P. Rangaiah, "An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes," *Comput. Chem. Eng.*, vol. 56, no. 9, pp. 155–173, 2013.

[8] S. Sudha, S. Baskar, S. M. J. Amali, and S. Krishnaswamy, "Protein structure prediction using diversity controlled self-adaptive differential evolution with local search," *Soft Comput.*, vol. 19, no. 6, pp. 1635–1646, 2014.

[9] V. V. D. Melo and G. L. C. Carosio, "Investigating multi-view differential evolution for solving constrained engineering design problems," *Expert Syst. Appli.*, vol. 40, no. 9, pp. 3370–3377, 2013.

[10] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Tran. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[11] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, 2011.

[12] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Tran. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, 2011.

[13] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, 2011.

[14] J. Q. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Tran. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

[15] W. J. Yu, M. Shen, W. N. Chen, Z. H. Zhan, Y. J. Gong, Y. Lin, O. Liu, and J. Zhang, "Differential evolution with two-level parameter adaptation," *IEEE Tran. Cybern.*, vol. 44, no. 7, pp. 1080–1099, 2014.

[16] A. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *in Proc. Australian Conf. Artif. Intell.*, Cairns, Dec. 2004, pp. 861–872.

[17] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*. Berlin, Germany: Springer-Verlag, 2009.

[18] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.

[19] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Tran. Cybern.*, vol. 43, no. 2, pp. 634–647, 2013.

[20] Y. W. Shang and Y. H. Qiu, "A note on the extended rosenbrock function," *Evol. Comput.*, vol. 14, no. 1, pp. 119–126, 2006.

[21] G. W. Corder and D. I. Foreman, *Nonparametric statistics for non-statisticians: a step-by-step approach*. Hoboken, New Jersey: John Wiley & Sons, 2009.

[22] J. J. Liang, B. Y. Qu, P. N. Suganthan, and Q. Chen, "Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization," Computational Intelligence Laboratory, Zhengzhou University, Tech. Rep., 2014.