Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters

Yong Wang, *Member, IEEE*, Zixing Cai, *Senior Member, IEEE*, and Qingfu Zhang, *Senior Member, IEEE*

***Abstract*—Trial vector generation strategies and control parameters have a significant influence on the performance of differential evolution (DE). This paper studies whether the performance of DE can be improved by combining several effective trial vector generation strategies with some suitable control parameter settings. A novel method, called *composite DE* (CoDE), has been proposed in this paper. This method uses three trial vector generation strategies and three control parameter settings. It randomly combines them to generate trial vectors. CoDE has been tested on all the CEC2005 contest test instances. Experimental results show that CoDE is very competitive.**

***Index Terms*—Differential evolution, trial vector generation strategy, control parameters, global numerical optimization**

I. Introduction

DIFFERENTIAL EVOLUTION (DE), proposed by Storn and Price in 1995 ([1], [2]), is a very popular evolutionary algorithm (EA) and exhibits remarkable performance in a wide variety of problems from diverse fields. Like other EAs, DE is a population-based stochastic search technique. It uses mutation, crossover, and selection operators at each generation to move its population towards the global optimum.

The DE performance mainly depends on two components. One is its trial vector generation strategy (i.e., mutation and crossover operators), and the other is its control parameters (i.e., population size *NP*, scaling factor *F*, and crossover control parameter C_r). In general, when using DE to solve optimization problems, we should firstly determine its trial vector generation strategy, and then tune the control parameters by a trial-and-error procedure. Since finding right parameter values in such a way is often very time-consuming, there has been an increasing interest in designing new DE variants with adaptive and self-adaptive control parameters. In adaptive parameter control [3], feedback from the search is exploited to adjust the parameter setting, and self-adaptive parameter control often encodes the parameters into the chromosome and evolves them from generation to generation. In addition to parameter adaptation [3], DE with strategy adaptation has also been studied by Qin *et al*. [4].

DE researchers have suggested many empirical guidelines for choosing trial vector generation strategies and control parameter settings during the last decade. It has been clear that some trial vector generation strategies are suitable for the global search [4] and some others are useful for rotated problems [5], and that some control parameter settings can speed up the convergence [6] and some other settings are effective for separable functions [7]. Undoubtedly, these experiences are very useful for improving the performance of DE. We have observed, however, that these experiences have not yet systematically exploited in DE algorithm design. This motivates us to study whether the DE performance can be improved by combining several trial vector generation strategies with several different control parameter settings, which have distinct advantages confirmed by other researchers' studies. Our work along this line has produced a *composite DE*, called CoDE. This proposed approach combines three well-studied trial vector generation strategies with three control parameter settings in a random way to generate trial vectors. CoDE has a very simple structure and thus is very easy to implement. Extensive experiments have been conducted in this paper to compare it with four other state-of-the-art DE variants and three other EAs on 25 commonly used CEC2005 contest test instances.

The rest of paper is organized as follows. Section II briefly introduces the basic DE operators. Section III reviews the related work on DE. The proposed approach is introduced in Section IV. Experimental results are reported in Section V. Finally, Section VI concludes this paper.

II. Differential Evolution (DE)

DE is for dealing with the continuous optimization problem. We suppose in this paper that the objective function to be minimized is $f(\vec{x})$, $\vec{x} = (x_1, \ldots, x_D) \in \Re^D$, and the feasible solution space is $\Omega = \prod_{i=1}^{D} [L_i, U_i]$.

At generation G=0, an initial population $\{\vec{x}_{i,0} = (x_{i,1,0}, x_{i,2,0}, \ldots, x_{i,D,0}),\ i = 1,2,\ldots,NP\}$ is randomly sampled from the feasible solution space, where *NP* is the population size.

At each generation *G*, DE creates a mutant vector $\vec{v}_{i,G} = (v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G})$ for each individual $\vec{x}_{i,G}$ (called a target vector) in the current population. The five widely used DE mutation operators are shown as follows:

- "rand/1":

$$\vec{v}_{i,G} = \vec{x}_{r1,G} + F \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}) \quad (1)$$

- "best/1":

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F \cdot (\vec{x}_{r1,G} - \vec{x}_{r2,G}) \quad (2)$$

Y. Wang and Z. Cai are with the School of Information Science and Engineering, Central South University, Changsha, 410083, P. R. China (e-mail: {ywang; zxcai}@csu.edu.cn).

Q. Zhang is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk).

- "current-to-best/1":

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r1,G} - \vec{x}_{r2,G}) \quad (3)$$

- "best/2":

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F \cdot (\vec{x}_{r1,G} - \vec{x}_{r2,G}) + F \cdot (\vec{x}_{r3,G} - \vec{x}_{r4,G}) \quad (4)$$

- "rand/2":

$$\vec{v}_{i,G} = \vec{x}_{r1,G} + F \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}) + F \cdot (\vec{x}_{r4,G} - \vec{x}_{r5,G}) \quad (5)$$

In the above equations, *r1*, *r2*, *r3*, *r4*, and *r5* 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 vectors. $\vec{x}_{best,G}$ is the best individual in the current population.

After mutation, DE performs a binomial crossover operator on $\vec{x}_{i,G}$ and $\vec{v}_{i,G}$ to generate a trial vector $\vec{u}_{i,G} = (u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G})$:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand_j(0,1) \le C_r \quad \text{or} \quad j = j_{rand}, \\ x_{i,j,G} & \text{otherwise}, \end{cases} \quad (6)$$

where $i = 1,2,\ldots,NP$, $j = 1,2,\ldots,D$, j_{rand} is a randomly chosen integer in [1,*D*], $rand_j(0,1)$ is a uniformly distributed random number between 0 and 1 which is generated for each *j*, and $C_r \in [0,1]$ is called the crossover control parameter. Due to the use of j_{rand}, the trial vector $\vec{u}_{i,G}$ differs from its target vector $\vec{x}_{i,G}$.

If the *j*th element $u_{i,j,G}$ of the trial vector $\vec{u}_{i,G}$ is infeasible (i.e., out of the boundary), it is reset as follows:

$$u_{i,j,G} = \begin{cases} \min\{U_j, 2L_j - u_{i,j,G}\} & \text{if} \quad u_{i,j,G} < L_j \\ \max\{L_j, 2U_j - u_{i,j,G}\} & \text{if} \quad u_{i,j,G} > U_j \end{cases} \quad (7)$$

The selection operator is performed to select the better one from the target vector $\vec{x}_{i,G}$ and the trial vector $\vec{u}_{i,G}$ to enter the next generation:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if } f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \quad (8)$$

III. Previous Work

Recognizing that the performance of DE depends on its trial vector generation strategies and its control parameters, researchers have proposed many DE variants during the past decade.

Some work mainly focuses on the trial vector generation strategies. Fan and Lampinen [8] proposed a trigonometric mutation operator to accelerate the DE convergence. Their mutation operator can be viewed as a local search operator, since it exploits the objective function value information and moves the new trial vector towards the direction provided by the best one of three individuals chosen for mutation. In order to balance the convergence speed and the search ability, they also introduced an extra parameter M_t for controlling the frequency of the use of the trigonometric mutation. Mezura-Montes *et al.* [9] proposed a novel mutation operator which incorporates the information of the best solution in the current population and the current parent to create a new trial vector. Feoktistov and Janaqi [10] classified mutation operators into four categories according to the way they use the objective function values. It has been observed that "current-to-best/1" strategy performs poorly on exploring the search space when solving multimodal problems [11]. Recently, much effort has been made to improve the performance of this strategy. Das *et al.* [5] improved the "current-to-best/1" strategy by introducing a local neighborhood model, in which each vector is mutated by using the best individual solution found so far in its small neighborhood. In addition, the local mutation model is combined with the global mutation model by a weight factor. Zhang and Sanderson [12] proposed the "current-to-*p*best/1" strategy. Instead of only adopting the best individual in the "current-to-best/1" strategy, their strategy also utilizes the information of other good solutions. Moreover, the recently generated inferior solutions are incorporated in this strategy. It is very interesting to note that some ideas of the above two approaches were inspired by particle swarm optimization (PSO).

Many attempts have also been made to improve the convergence speed and robustness of DE by tuning the control parameters such as the population size *NP*, the scaling factor *F*, and the crossover control parameter C_r. Storn and Price [2] argued that these three control parameters are not difficult to set for obtaining good performance. They suggested that *NP* should be between *5D* and *10D*, *F* should be 0.5 as a good initial choice and the value of *F* smaller than 0.4 or larger than 1.0 will lead to performance degradation, and C_r can be set to 0.1 or 0.9. In contrast, Gämperle *et al.* [13] showed that the performance of DE is very sensitive to the setting of the control parameters based on their experiments on Sphere, Rosenbrock, and Rastrigin functions. They suggested that *NP* should be between *3D* and *8D*. They argued that the value of *F* should not be smaller than a problem-dependent threshold value in order to prevent premature convergence, and that if *F* is larger than 1.0, the convergence speed will decrease. So, they suggested that a good initial choice of *F* is 0.6. A suggested value for C_r is between 0.3 and 0.9 in [13]. Ronkkonen *et al.* [7] suggested that *NP* should be between *2D* and *4D*, *F* should be chosen from the range [0.4, 0.95] with *F*=0.9 being a good trade-off between convergence speed and robustness, and C_r should be between 0.0 and 0.2 for separable functions and between 0.9 and 1.0 for multimodal and non-separable functions. Clearly, these researchers agreed that *F* should be in the range of [0.4, 1.0], and that C_r should be either close to 1.0 or 0.0 depending on the characteristics of problems. However, there is no agreement on the setting of *NP*.

Das *et al.* [14] introduced two schemes to adapt the scaling factor *F* in DE. One scheme varies *F* in a random manner, and the other one linearly reduces the value of *F* from a preset maximal value to a minimal one. Liu and Lampinen [15] proposed a fuzzy adaptive DE (FADE), which uses a fuzzy knowledge-based system to dynamically adjust *F* and C_r. In their method, the mean square roots of differences of

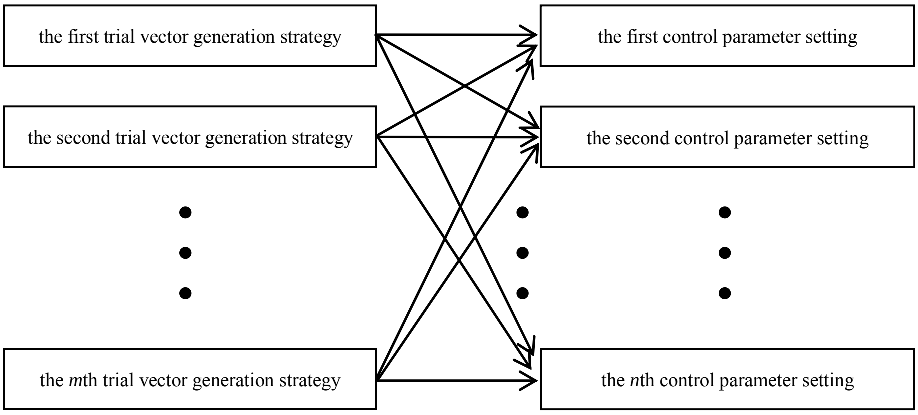

Fig. 1 Illustration of combining trial vector generation strategies with control parameter settings.

the function values and the population members during the successive generations are used as the inputs of the fuzzy logic controller, and F and C_r are the outputs. Brest *et al.* [16] proposed a self-adaptive DE (jDE), in which both F and C_r are applied at individual level. During the evolution, the new F takes a value from 0.1 to 0.9 in a random manner with a probability τ_1, the new C_r takes a value from 0.0 to 1.0 in a random manner with a probability τ_2, and both of them are obtained before the mutation is executed. In the JADE proposed by Zhang and Sanderson [12], a normal distribution and a Cauchy distribution are utilized to generate F and C_r for each target vector, respectively. JADE extracts information from the recent successful F and C_r and uses such information for generating new F and C_r. Besides the adaptation of the control parameters F and C_r, Teo [17] investigated the population sizing problem via self-adaptation and proposed two different approaches, one adopts an absolute encoding strategy for *NP*, and the other adopts a relative encoding strategy for *NP*.

Unlike the above methods, SaDE, proposed by Qin *et al.* [4], adaptively adjusts its trial vector generation strategies and control parameters simultaneously by learning from the previous search. During the late stage of this paper revision, we were aware that Mallipeddi *et al.* [18] very recently proposed an ensemble of trial vector generation strategies and control parameters of DE (EPSDE). EPSDE includes a pool of distinct trial vector generation strategies and a pool of values for the control parameters F and C_r.

IV. Composite DE (CoDE)

As pointed out in Section III, the characteristics of the trial vector generation strategies and the control parameters of DE have been extensively investigated, and that some prior knowledge has been obtained during the last decade. Such prior knowledge could be used for designing more effective and robust DE variants. We also observed that in most DE variants including adaptive and self-adaptive DE variants, only one trial vector generation strategy and only one control parameter setting are employed at each generation for each target vector. As a result, the search ability of these algorithms could be limited.

Based on the above considerations, we propose a novel method, called *composite DE* (CoDE), the primary idea of which is to randomly combine several trial vector generation strategies with a number of control parameter settings at each generation to create new trial vectors. The above idea is illustrated in Fig. 1.

In general, we expect that the chosen trial vector generation strategies and control parameter settings show distinct advantages and, therefore, they can be effectively combined to solve different kinds of problems. In this paper, we choose three trial vector generation strategies and three control parameter settings to constitute the strategy candidate pool and the parameter candidate pool, respectively. Thus the parameters m and n in Fig. 1 are equal to 3. The three selected trial vector generation strategies are:

- "rand/1/bin",
- "rand/2/bin", and
- "current-to-rand/1".

Note that in the "current-to-rand/1" strategy, the binominal crossover operator is not applied. The three control parameter settings are:

- [F=1.0, Cr=0.1],
- [F=1.0, Cr=0.9], and
- [F=0.8, Cr=0.2].

The above strategies and parameter settings are frequently used in many DE variants and their properties have been well studied. The three strategies are shown in the equations at the bottom of the next page, where *rand* denotes a uniformly distributed random number between 0 and 1. In order to further improve the search ability of the "rand/2/bin" strategy, the first scaling factor F in the "rand/2" mutation operator is randomly chosen from 0 to 1 in this paper.

At each generation, each trial vector generation strategy in the strategy candidate pool is used to create a new trial vector with a control parameter setting randomly chosen from the parameter candidate pool. Thus, three trial vectors are generated for each target vector. Then the best one enters the next generation if it is better than its target vector. The pseudocode of CoDE is presented in Fig. 2.

To the best of our knowledge, EPSDE [18] was the first

Input: *NP*: the number of individuals at each generation, i.e, the population size.
Max_FES: maximum number of function evaluation evaluations.
the strategy candidate pool: "rand/1/bin", "rand/2/bin", and "current-to-rand/1".
the parameter candidate pool: [F=1.0, C_r=0.1], [F=1.0, C_r=0.9], and [F=0.8, C_r=0.2].
(1) G=0;
(2) Generate an initial population $P_0 = \{\vec{x}_{1,0},\dots,\vec{x}_{NP,0}\}$ by uniformly and randomly sampling from the feasible solution space;
(3) Evaluate the objective function values $f(\vec{x}_{1,0}),\dots,f(\vec{x}_{NP,0})$;
(4) *FES*=*NP*;
(5) **while** *FES*<*Max_FES* **do**
(6) $P_{G+1} = \emptyset$;
(7) **for** i=1:*NP* **do**
(8) Use the three strategies, each with a control parameter setting randomly selected from the parameter pool, to generate three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$ for the target vector $\vec{x}_{i,G}$;
(9) Evaluate the objective function values of the three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$;
(10) Choose the best trial vector (denoted as $\vec{u}^*_{i,G}$) from the three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$;
(11) $P_{G+1} = P_{G+1} \cup select(\vec{x}_{i,G}, \vec{u}^*_{i,G})$;
(12) *FES*=*FES*+3;
(13) **end for**
(14) G=G+1;
(15) **end while**
Output: the individual with the smallest objective function value in the population.

Fig. 2. Pseudocode of CoDE

attempt to provide a systematic framework for combining different trial vector generation strategies with different control parameter settings. Mallipeddi and Suganthan proposed another version of EPSDE in [26]. CoDE differs from EPSDE [18] in the following major aspects:

1) CoDE selects trial vector generation strategies and control parameter settings based on experimental results reported in the literature. Each strategy in CoDE is coupled with a randomly chosen parameter setting for generating a new solution. In contrast, EPSDE learns good combinations from evolution. If a combination performs better in the previous search, it will have more chances to be used in the further search in EPSDE. CoDE aims at making good use of other researchers' experiences whereas EPSDE focuses on learning good combinations.
2) The strategy candidate pool of CoDE is different from that of EPSDE. In CoDE, three fixed control parameter settings are used. In EPSDE, however, F is chosen from 0.4 to 0.9 with step-size 0.1 and C_r is chosen from 0.1 to 0.9 with step-size 0.1. CoDE chooses its strategies and parameter settings based on other researchers' studies.
3) In CoDE, three trial vectors are generated for each target vector. However, as in the conventional DE only one trial vector is produced for each target vector in EPSDE.

Next, we discuss the properties of the strategy candidate pool and the parameter candidate pool.

A. Basic Properties of the Strategy Candidate Pool

The "rand/1/bin" strategy is the most commonly used strategy in the literature. In this strategy, all vectors for mutation are selected from the population at random and, consequently, it has no bias to any special search directions and chooses new search directions in a random manner. In the "rand/2/bin" strategy, two difference vectors are added to the base vector, which might lead to better perturbation than the strategies with only one difference vector [4]. Moreover, it can generate more different trial vectors than the "rand/1/bin" strategy. After mutation, the "current-to-rand/1" strategy uses the rotation-invariant arithmetic crossover

"rand/1/bin":

$$u_{i,j,G} = \begin{cases} x_{r1,j,G} + F\cdot(x_{r2,j,G} - x_{r3,j,G}), & \text{if } rand < C_r \quad or \quad j = j_{rand} \\ x_{i,j,G} & otherwise \end{cases}$$

"rand/2/bin":

$$u_{i,j,G} = \begin{cases} x_{r1,j,G} + F\cdot(x_{r2,j,G} - x_{r3,j,G}) + F\cdot(x_{r4,j,G} - x_{r5,j,G}) & \text{if } rand < C_r \quad or \quad j = j_{rand} \\ x_{i,j,G} & otherwise \end{cases}$$

"current-to-rand/1":

$$\vec{u}_{i,G} = \vec{x}_{i,G} + rand\cdot(\vec{x}_{r1,G} - \vec{x}_{i,G}) + F\cdot(\vec{x}_{r2,G} - \vec{x}_{r3,G})$$

TABLE I
EXPERIMENTAL RESULTS OF JADE, JDE, SADE, EPSDE, AND CODE OVER 25 INDEPENDENT RUNS ON 25 TEST FUNCTIONS OF 30 VARIABLES WITH 300,000 FES. "MEAN ERROR" AND "STD DEV" INDICATE THE AVERAGE AND STANDARD DEVIATION OF THE FUNCTION ERROR VALUES OBTAINED IN 25 RUNS, RESPECTIVELY. WILCOXON'S RANK SUM TEST AT A 0.05 SIGNIFICANCE LEVEL IS PERFORMED BETWEEN CODE AND EACH OF JADE, JDE, SADE, AND EPSDE.

Function		JADE Mean Error±Std Dev	jDE Mean Error±Std Dev	SaDE Mean Error±Std Dev	EPSDE Mean Error±Std Dev	CoDE Mean Error±Std Dev
Unimodal Functions	F_1	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	F_2	1.07E-28±1.00E-28+	1.11E-06±1.96E-06−	8.26E-06±1.65E-05−	4.23E-26±4.07E-26+	1.69E-15±3.95E-15
	F_3	8.42E+03±7.26E+03+	1.98E+05±1.10E+05−	4.27E+05±2.08E+05−	8.74E+05±3.28E+06−	1.05E+05±6.25E+04
	F_4	1.73E-16±5.43E-16+	4.40E-02±1.26E-01−	1.77E+02±2.67E+02−	3.49E+02±2.23E+03−	5.81E-03±1.38E-02
	F_5	8.59E-08±5.23E-07+	5.11E+02±4.40E+02−	3.25E+03±5.90E+02−	1.40E+03±7.12E+02−	3.31E+02±3.44E+02
Basic Multimodal Functions	F_6	1.02E+01±2.96E+01−	2.35E+01±2.50E+01−	5.31E+01±3.25E+01−	6.38E-01±1.49E+00−	1.60E-01±7.85E-01
	F_7	8.07E-03±7.42E-03≈	1.18E-02±7.78E-03−	1.57E-02±1.38E-02−	1.77E-02±1.34E-02−	7.46E-03±8.55E-03
	F_8	2.09E+01±1.68E-01−	2.09E+01±4.86E-02−	2.09E+01±4.95E-02−	2.09E+01±5.81E-02−	2.01E+01±1.41E-01
	F_9	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	2.39E-01±4.33E-01−	3.98E-02±1.99E-01−	0.00E+00±0.00E+00
	F_{10}	2.41E+01±4.61E+00+	5.54E+01±8.46E+00−	4.72E+01±1.01E+01−	5.36E+01±3.03E+01−	4.15E+01±1.16E+01
	F_{11}	2.53E+01±1.65E+00−	2.79E+01±1.61E+00−	1.65E+01±2.42E+00−	3.56E+01±3.88E+00−	1.18E+01±3.40E+00
	F_{12}	6.15E+03±4.79E+03−	8.63E+03±8.31E+03−	3.02E+03±2.33E+03≈	3.58E+04±7.05E+03−	3.05E+03±3.80E+03
Expanded Multimodal Functions	F_{13}	1.49E+00±1.09E-01≈	1.66E+00±1.35E-01−	3.94E+00±2.81E-01−	1.94E+00±1.46E-01−	1.57E+00±3.27E-01
	F_{14}	1.23E+01±3.11E-01≈	1.30E+01±2.00E-01−	1.26E+01±2.83E-01−	1.35E+01±2.09E-01−	1.23E+01±4.81E-01
Hybrid Composition Functions	F_{15}	3.51E+02±1.28E+02≈	3.77E+02±8.02E+01≈	3.76E+02±7.83E+01≈	2.12E+02±1.98E+01+	3.88E+02±6.85E+01
	F_{16}	1.01E+02±1.24E+02−	7.94E+01±2.96E+01−	8.57E+01±6.94E+01≈	1.22E+02±9.19E+01−	7.37E+01±5.13E+01
	F_{17}	1.47E+02±1.33E+02−	1.37E+02±3.80E+01−	7.83E+01±3.76E+01−	1.69E+02±1.02E+02−	6.67E+01±2.12E+01
	F_{18}	9.04E+02±1.03E+00≈	9.04E+02±1.08E+01≈	8.68E+02±6.23E+01≈	8.20E+02±3.35E+00+	9.04E+02±1.04E+00
	F_{19}	9.04E+02±8.40E-01≈	9.04E+02±1.11E+00≈	8.74E+02±6.22E+01≈	8.21E+02±3.35E+00+	9.04E+02±9.42E-01
	F_{20}	9.04E+02±8.47E-01≈	9.04E+02±1.10E+00≈	8.78E+02±6.03E+01≈	8.22E+02±4.17E+00+	9.04E+02±9.01E-01
	F_{21}	5.00E+02±4.67E-13≈	5.00E+02±4.80E-13≈	5.52E+02±1.82E+02−	8.33E+02±1.00E+02−	5.00E+02±4.88E-13
	F_{22}	8.66E+02±1.91E+01≈	8.75E+02±1.91E+01−	9.36E+02±1.83E+01−	5.07E+02±7.26E+00+	8.63E+02±2.43E+01
	F_{23}	5.50E+02±8.05E+01−	5.34E+02±2.77E-04≈	5.34E+02±3.57E-03≈	8.58E+02±6.82E+01−	5.34E+02±4.12E-04
	F_{24}	2.00E+02±2.85E-14≈	2.00E+02±2.85E-14≈	2.00E+02±6.20E-13≈	2.13E+02±1.52E+00−	2.00E+02±2.85E-14
	F_{25}	2.11E+02±7.92E-01≈	2.11E+02±7.32E-01≈	2.14E+02±2.00E+00−	2.13E+02±2.55E+00−	2.11E+02±9.02E-01
−		7	15	16	18	
+		5	0	0	6	
≈		13	10	9	1	

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoDE, respectively.

rather than the binomial crossover, to generate the trial vector ([5], [19]). As a result, this strategy is rotation-invariant and suitable for rotated problems. The arithmetic crossover in this strategy linearly combines the mutant vector with the target vector to generate the trial vector as follows:

$$\vec{u}_{i,G} = \vec{x}_{i,G} + rand \cdot (\vec{v}_{i,G} - \vec{x}_{i,G}) \tag{9}$$

where *rand* is a uniformly distributed random number between 0 and 1. Note that for the arithmetic crossover the crossover control parameter C_r in DE is not needed.

Some strategies such as "best/1/bin" strategy and "best/2/bin" strategy utilize the information of the best individual found so far, they might not be very effective when solving multimodal problems. For this reason, we do not use these strategies in this paper.

B. Basic Properties of the Parameter Candidate Pool

In general, a large value of F can make the mutant vectors distribute widely in the search space and can increase the population diversity. In contrast, a low value of F makes the search focus on neighborhoods of the current solutions, and thus it can speed up the convergence.

A large value of C_r can make the trial vector very different from the target vector, since the trial vector inherits little information from the target vector. Therefore, the diversity of the offspring population can be encouraged. A small value of C_r is very suitable for separable problems, since in this case the trial vector may be different from the target vector by only one parameter and, as a result, each parameter is optimized independently.

In summary, the selected strategies and parameter settings exhibit distinct advantages. Therefore, they are expected to complement one another for solving optimization problems of different characteristics. Actually, each of the three parameter settings has the same property for the three strategies. For instance, the first control parameter setting, [F=1.0, C_r=0.1], is for dealing with separable problems when

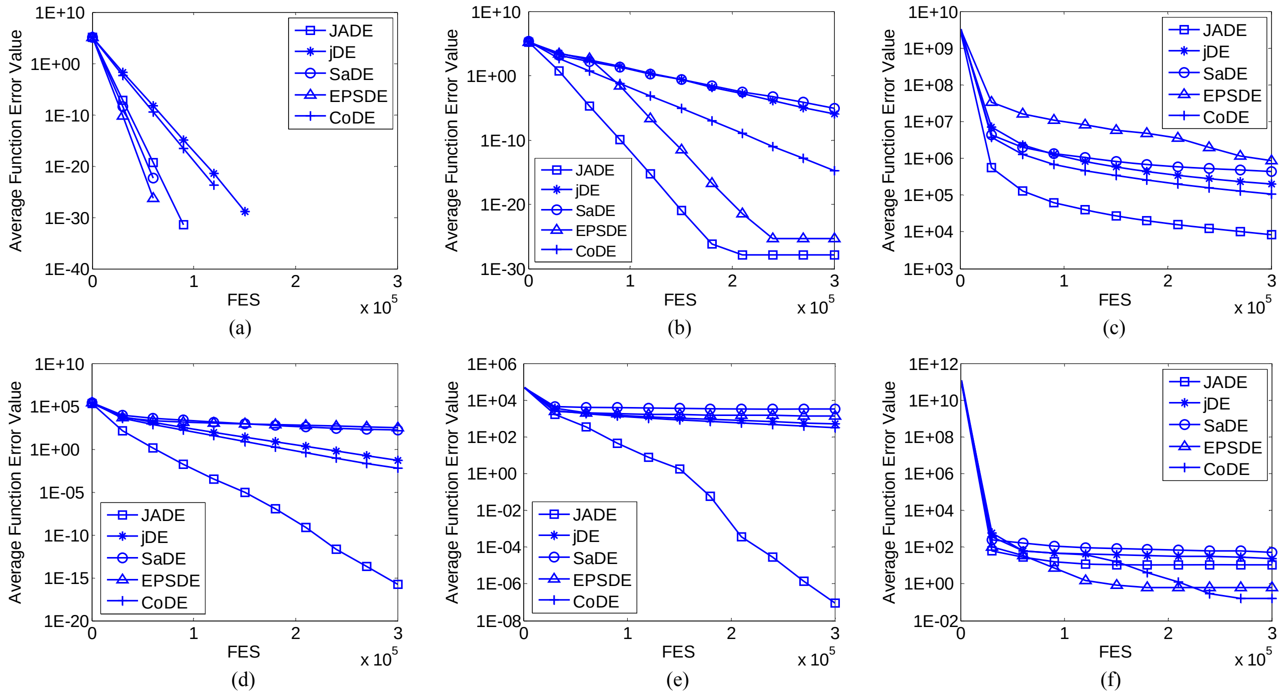

Fig. 3. Evolution of the mean function error values derived from JADE, jDE, SaDE, EPSDE, and CoDE versus the number of FES on six test functions. (a) F_1. (b) F_2. (c) F_3. (d) F_4. (e) F_5. (f) F_6.

combined with the three strategies, the second control parameter setting, [F=1.0, C_r=0.9], is mainly to maintain the population diversity and to make the three strategies more powerful in global exploration, and the last control parameter setting, [F=0.8, C_r=0.2], encourages the exploitation of the three strategies in the search space and thus accelerates the convergence speed of the population.

V. Experimental Study

25 test instances proposed in the CEC2005 special session on real-parameter optimization were used to study the performance of the proposed CoDE. A detailed description of these test instances can be found in [20]. These 25 test instances can be divided into four classes:

1) unimodal functions F_1-F_5,
2) basic multimodal functions F_6-F_{12},
3) expanded multimodal functions F_{13}-F_{14}, and
4) hybrid composition functions F_{15}-F_{25}.

The number of decision variables, D, was set to 30 for all the 25 test functions. For each algorithm and each test function, 25 independent runs were conducted with 300,000 function evaluations (FES) as the termination criterion. The population size in CoDE was set to 30.

In our experimental studies, the average and standard deviation of the function error value ($f(\vec{x})-f(\vec{x}^*)$) were recorded for measuring the performance of each algorithm, where $\vec{x}$ is the best solution found by the algorithm in a run and $\vec{x}^*$ is the global optimum of the test function. CoDE was compared with four other DE variants and three non-DE approaches. In order to have statistically sound conclusions, Wilcoxon's rank sum test at a 0.05 significance level was conducted on the experimental results.

A. Comparison with Four State-of-the-art DE

CoDE was compared with four other state-of-the-art DE variants, i.e., JADE [12], jDE [16], SaDE [4], and EPSDE [18]. In JADE, jDE, and SaDE, the control parameters F and C_r were self-adapted during the evolution. In our experiments, we used the same parameter settings for these four methods as in their original papers. The number of FES in all these methods was set to 300,000, as the same as in CoDE.

The experimental results are given in Table I. All the results are obtained from 25 independent runs. The last three rows of Table I summarize the experimental results.

1) Unimodal functions F_1-F_5. Clearly, JADE is the best among the five methods on these five unimodal functions. It outperforms CoDE on four test functions (i.e., F_2-F_5). The outstanding performance of JADE should be due to its greedy mutation strategy ("current-to-pbest/1" strategy), which leads to very fast convergence. CoDE is the second best. It performs better than jDE, SaDE, and EPSDE on 4, 4, and 3 test functions, respectively. jDE and SaDE cannot outperform CoDE on any test function and EPSDE surpasses CoDE on one test function (i.e., F_2).

2) Basic multimodal functions F_6-F_{12}. On these seven test functions, CoDE is significantly better than JADE, jDE, SaDE, and EPSDE on 4, 6, 6, and 7 test functions, respectively. JADE outperforms CoDE on one test function, and jDE, SaDE, and EPSDE cannot be significantly better than CoDE on any test function. Thus, CoDE is the winner on these seven test functions. This can be because CoDE could balance exploration and exploitation on these test functions by combining different trial vector generation

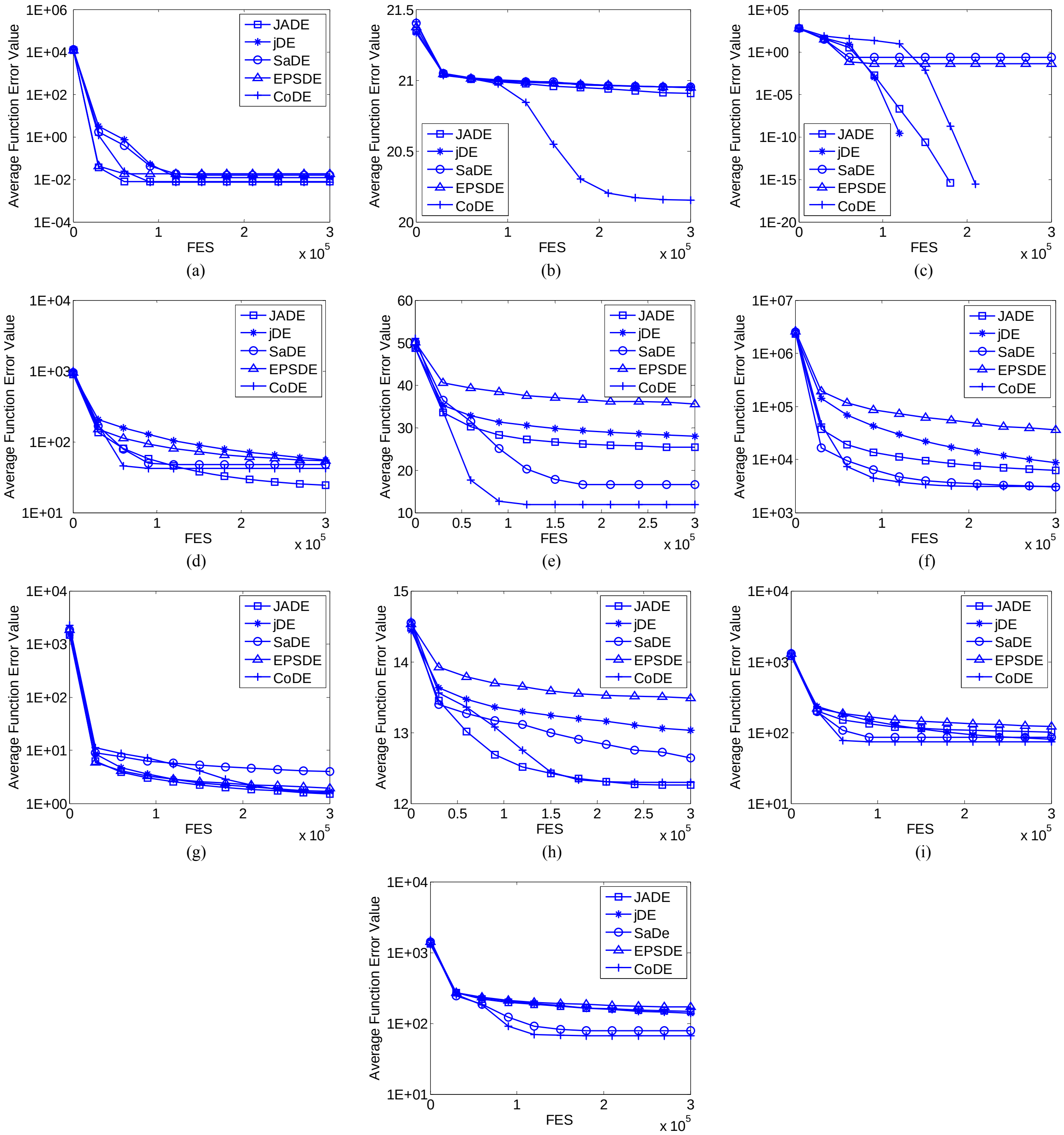

Fig. 4. Evolution of the mean function error values derived from JADE, jDE, SaDE, EPSDE, and CoDE versus the number of FES on ten test functions. (a) F_7. (b) F_8. (c) F_9. (d) F_{10}. (e) F_{11}. (f) F_{12}. (g) F_{13}. (h) F_{14}. (i) F_{16}. (j) F_{17}.

strategies with different control parameter settings.

3) Expanded multimodal functions F_{13}-F_{14}. On these two test functions, CoDE and JADE exhibit similar performance and outperform three other methods.

4) Hybrid composition functions F_{15}-F_{25}. These test functions are much harder than others since each of them is composed of 10 sub-functions. No method can reduce the average function error value below 10 on any test function. Overall, the performance of CoDE is better than that of the four competitors. It outperforms JADE, jDE, SaDE, and EPSDE on 3, 3, 4, and 6 test functions, respectively. In contrast, JADE, jDE, and SaDE cannot perform better than CoDE even on one test function. It is interesting to note that for test functions F_{15}, F_{18}, F_{19}, F_{20}, and F_{22}, EPSDE is significantly better than four others according to the Wilcoxon's rank sum test.

In summary, CoDE is the best among the five methods in comparison on basic multimodal functions, expanded multimodal functions, and hybrid composition functions. It is the second best on unimodal functions. The last three rows in

TABLE II
EXPERIMENTAL RESULTS OF CLPSO, CMA-ES, GL-25, AND CODE OVER 25 INDEPENDENT RUNS ON 25 TEST FUNCTIONS OF 30 VARIABLES WITH 300,000 FES. "MEAN ERROR" AND "STD DEV" INDICATE THE AVERAGE AND STANDARD DEVIATION OF THE FUNCTION ERROR VALUES OBTAINED IN 25 RUNS, RESPECTIVELY. WILCOXON'S RANK SUM TEST AT A 0.05 SIGNIFICANCE LEVEL IS PERFORMED BETWEEN CODE AND EACH OF CLPSO, CMA-ES, AND GL-25.

Function		CLPSO Mean Error±Std Dev	CMA-ES Mean Error±Std Dev	GL-25 Mean Error±Std Dev	CoDE Mean Error±Std Dev
Unimodal Functions	F_1	0.00E+00±0.00E+00≈	1.58E-25±3.35E-26－	5.60E-27±1.76E-26－	0.00E+00±0.00E+00
	F_2	8.40E+02±1.90E+02－	1.12E-24±2.93E-25＋	4.04E+01±6.28E+01－	1.69E-15±3.95E-15
	F_3	1.42E+07±4.19E+06－	5.54E-21±1.69E-21＋	2.19E+06±1.08E+06－	1.05E+05±6.25E+04
	F_4	6.99E+03±1.73E+03－	9.15E+05±2.16E+06－	9.07E+02±4.25E+02－	5.81E-03±1.38E-02
	F_5	3.86E+03±4.35E+02－	2.77E-10±5.04E-11＋	2.51E+03±1.96E+02－	3.31E+02±3.44E+02
Basic Multimodal Functions	F_6	4.16E+00±3.48E+00－	4.78E-01±1.32E+00－	2.15E+01±1.17E+00－	1.60E-01±7.85E-01
	F_7	4.51E-01±8.47E-02－	1.82E-03±4.33E-03＋	2.78E-02±3.62E-02－	7.46E-03±8.55E-03
	F_8	2.09E+01±4.41E-02－	2.03E+01±5.72E-01－	2.09E+01±5.94E-02－	2.01E+01±1.41E-01
	F_9	0.00E+00±0.00E+00≈	4.45E+02±7.12E+01－	2.45E+01±7.35E+00－	0.00E+00±0.00E+00
	F_{10}	1.04E+02±1.53E+01－	4.63E+01±1.16E+01≈	1.42E+02±6.45E+01－	4.15E+01±1.16E+01
	F_{11}	2.60E+01±1.63E+00－	7.11E+00±2.14E+00＋	3.27E+01±7.79E+00－	1.18E+01±3.40E+00
	F_{12}	1.79E+04±5.24E+03－	1.26E+04±1.74E+04－	6.53E+04±4.69E+04－	3.05E+03±3.80E+03
Expanded Multimodal Functions	F_{13}	2.06E+00±2.15E-01－	3.43E+00±7.60E-01－	6.23E+00±4.88E+00－	1.57E+00±3.27E-01
	F_{14}	1.28E+01±2.48E-01－	1.47E+01±3.31E-01－	1.31E+01±1.84E-01－	1.23E+01±4.81E-01
Hybrid Composition Functions	F_{15}	5.77E+01±2.76E+01＋	5.55E+02±3.32E+02－	3.04E+02±1.99E+01＋	3.88E+02±6.85E+01
	F_{16}	1.74E+02±2.82E+01－	2.98E+02±2.08E+02－	1.32E+02±7.60E+01－	7.37E+01±5.13E+01
	F_{17}	2.46E+02±4.81E+01－	4.43E+02±3.34E+02－	1.61E+02±6.80E+01－	6.67E+01±2.12E+01
	F_{18}	9.13E+02±1.42E+00－	9.04E+02±3.01E-01≈	9.07E+02±1.48E+00－	9.04E+02±1.04E+00
	F_{19}	9.14E+02±1.45E+00－	9.16E+02±6.03E+01－	9.06E+02±1.24E+00－	9.04E+02±9.42E-01
	F_{20}	9.14E+02±3.62E+00－	9.04E+02±2.71E-01≈	9.07E+02±1.35E+00－	9.04E+02±9.01E-01
	F_{21}	5.00E+02±3.39E-13－	5.00E+02±2.68E-12－	5.00E+02±4.83E-13－	5.00E+02±4.88E-13
	F_{22}	9.72E+02±1.20E+01－	8.26E+02±1.46E+01＋	9.28E+02±7.04E+01－	8.63E+02±2.43E+01
	F_{23}	5.34E+02±2.19E-04≈	5.36E+02±5.44E+00－	5.34E+02±4.66E-04≈	5.34E+02±4.12E-04
	F_{24}	2.00E+02±1.49E-12－	2.12E+02±6.00E+01－	2.00E+02±5.52E-11－	2.00E+02±2.85E-14
	F_{25}	2.00E+02±1.96E+00＋	2.07E+02±6.07E+00≈	2.17E+02±1.36E-01－	2.11E+02±9.02E-01
－		20	15	23	
＋		2	6	1	
≈		3	4	1	

"－", "＋", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoDE, respectively.

Table I indicate that, overall, CoDE is better than the four competitors. The evolution of the mean function error values derived from JADE, jDE, SaDE, EPSDE, and CoDE versus the number of FES is plotted in Figs. 3-4 for some typical test functions.

B. Comparison with CLPSO, CMA-ES, and GL-25

CoDE was also compared with three non-DE approaches, namely, CLPSO [21], CMA-ES [22], and GL-25 [23]. CLPSO is proposed by Liang *et al*. [21]. In CLPSO, a particle uses the personal historical best information of all the particles to update its velocity. CMA-ES, proposed by Hansen and Ostermeier [22], is a very efficient and famous evolution strategy (ES). There are actually several variants of CMA-ES, such as the restart CMA-ES [24]. In this paper, the standard CMA-ES [22] is used for comparison. GL-25, proposed by Garcia-Martinez *et al*. [23], is a hybrid real-coded genetic algorithm (RCGA) which combines the global and local search. In our experiments, the parameter settings of these three methods were the same as in their original papers. As in the experiments in Section V-A, the number of FES in all these methods was 300,000, and each method was run 25 times on each test function. Table II summarizes the experimental results.

Overall, CoDE significantly outperforms CLPSO, CMA-ES, and GL-25. In fact, CoDE performs better than CLPSO, CMA-ES, and GL-25 on 20, 15, and 23 out of 25 test functions, respectively. CLPSO beats CoDE on 2 test functions, CMA-ES is better than CoDE on 6 test functions, and GL-25 outperforms CoDE on only one test function.

C. Random Selection of the Control Parameter Settings Versus Deterministic Selection of the Control Parameter Settings

Each trial vector generation strategy (i.e., "rand/1/bin", "rand/2/bin", or "current-to-rand/1") in CoDE randomly

TABLE III
EXPERIMENTAL RESULTS OF CODE VARIANTS WITH FIXED PARAMETER SETTINGS AND CODE OVER 25 INDEPENDENT RUNS FOR 25 TEST FUNCTIONS OF 30 VARIABLES WITH 300,000 FES. "MEAN ERROR" AND "STD DEV" INDICATE THE AVERAGE AND STANDARD DEVIATION OF THE FUNCTION ERROR VALUES OBTAINED IN 25 RUNS, RESPECTIVELY. WILCOXON'S RANK SUM TEST AT A 0.05 SIGNIFICANCE LEVEL IS PERFORMED BETWEEN CODE VARIANTS WITH FIXED PARAMETER SETTINGS AND CODE.

Function		CoDE-132 Mean Error±Std Dev	CoDE-212 Mean Error±Std Dev	CoDE-312 Mean Error±Std Dev	CoDE Mean Error±Std Dev
Unimodal Functions	F_1	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	F_2	1.64E-14±3.49E-14－	4.43E-13±9.94E-13－	1.35E-14±2.79E-14－	1.69E-15±3.95E-15
	F_3	1.39E+05±9.52E+04－	1.24E+05±7.09E+04－	1.28E+05±6.76E+04－	1.05E+05±6.25E+04
	F_4	2.32E-03±5.35E-03＋	3.11E-02±1.04E-01－	1.92E-03±5.05E-03＋	5.81E-03±1.38E-02
	F_5	4.21E+02±3.83E+02－	3.23E+02±3.79E+02≈	3.26E+02±2.96E+02≈	3.31E+02±3.44E+02
Basic Multimodal Functions	F_6	1.60E-01±7.85E-01≈	1.60E-01±7.85E-01≈	1.60E-01±7.85E-01≈	1.60E-01±7.85E-01
	F_7	9.23E-03±9.13E-03≈	9.35E-03±1.03E-02≈	8.64E-03±8.51E-03≈	7.46E-03±8.55E-03
	F_8	2.01E+01±1.07E-02≈	2.02E+01±1.62E-01－	2.01E+01±1.19E-02≈	2.01E+01±1.41E-01
	F_9	0.00E+00±0.00E+00≈	8.05E-01±8.79E-01－	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	F_{10}	4.07E+01±1.12E+01≈	4.44E+01±1.36E+01≈	4.32E+01±1.84E+01≈	4.15E+01±1.16E+01
	F_{11}	1.21E+01±3.16E+00≈	1.19E+01±3.22E+00≈	1.17E+01±3.23E+00≈	1.18E+01±3.40E+00
	F_{12}	2.59E+03±3.37+03≈	2.93E+03±4.67E+03≈	2.81E+03±3.06E+03≈	3.05E+03±3.80E+03
Expanded Multimodal Functions	F_{13}	1.52E+00±2.83E-01≈	1.76E+00±3.86E-01－	1.52E+00±3.07E-01≈	1.57E+00±3.27E-01
	F_{14}	1.23E+01±5.02E-01≈	1.25E+01±4.48E-01－	1.23E+01±4.45E-01≈	1.23E+01±4.81E-01
Hybrid Composition Functions	F_{15}	3.95E+02±6.09E+01≈	4.02E+02±5.31E+01≈	4.02E+02±6.35E+01≈	3.88E+02±6.85E+01
	F_{16}	7.19E+01±3.72E+01≈	7.78E+01±4.78E+01－	7.49E+01±3.92E+01－	7.37E+01±5.13E+01
	F_{17}	7.12E+02±3.87E+01－	7.75E+01±4.14E+01－	8.13E+01±5.66E+01－	6.67E+01±2.12E+01
	F_{18}	9.04E+02±8.70E-01≈	9.04E+02±9.60E-01≈	9.04E+02±8.97E-01≈	9.04E+02±1.04E+00
	F_{19}	9.04E+02±1.10E+00≈	9.04E+02±1.12E+00≈	9.04E+02±6.09E-01≈	9.04E+02±9.42E-01
	F_{20}	9.04E+02±9.15E-01≈	9.04E+02±1.09E+00≈	9.04E+02±1.08E+00≈	9.04E+02±9.01E-01
	F_{21}	5.00E+02±4.75E-13≈	5.00E+02±5.54E-13≈	5.00E+02±4.62E-13≈	5.00E+02±4.88E-13
	F_{22}	8.67E+02±2.37E+01≈	8.66E+02±2.85E+01≈	8.70E+02±2.42E+01－	8.63E+02±2.43E+01
	F_{23}	5.34E+02±3.28E-04≈	5.34E+02±4.09E-04≈	5.34E+02±4.07E-04≈	5.34E+02±4.12E-04
	F_{24}	2.00E+02±2.85E-14≈	2.00E+02±2.85E-14≈	2.00E+02±2.85E-14≈	2.00E+02±2.85E-14
	F_{25}	2.11E+02±8.51E-01≈	2.11E+02±8.64E-01≈	2.11E+02±7.86E-01≈	2.11E+02±9.02E-01
－		4	9	5	
＋		1	0	1	
≈		20	16	19	

"－", "＋", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoDE, respectively.

selects a control parameter setting from the three predetermined parameter groups for generating a trial vector. One would like to know what if each strategy uses a fixed control parameter setting in the search. To address this issue, we tested 27 CoDE variants. These variants were the same as the original CoDE in Fig. 2 except that each trial vector generation strategy used a fixed control parameter setting selected from the three parameter groups.

For each variant, 25 independent runs were carried out on 25 test functions. Due to the page limit, we only report the experimental results of the three best variants among all the 27 ones in Table III. The three best ones are:

- CoDE-132: "rand/1/bin" uses the first parameter setting, "rand/2/bin" uses the third one, and "current-to-rand/1" uses the second one.
- CoDE-212: "rand/1/bin" uses the second parameter setting, "rand/2/bin" uses the first one, and "current-to-rand/1" uses the second one.
- CoDE-312: "rand/1/bin" uses the third parameter setting, "rand/2/bin" uses the first one, and "current-to-rand/1" uses the second one.

Table III shows that CoDE is significantly better than CoDE-132, CoDE-212 and CoDE-312 on 4, 9, and 5 out of 25 test functions, respectively. These three variants win CoDE on 1, 0, and 1 test function, respectively. Therefore, we can conclude that the use of random control parameter settings in CoDE does make the search more effective. This should be because random selection of the control parameter settings can increase the search diversity.

D. Random Selection of the Control Parameter Settings Versus Adpative Selection of the Control Parameter Settings

In CoDE, each control parameter setting has the same probability to be used. Therefore, it can be regarded as an alternating heuristic [25]. Recently, some adaptive

TABLE IV
EXPERIMENTAL RESULTS OF THE ADAPTIVE CODE AND CODE OVER 25 INDEPENDENT RUNS FOR 25 TEST FUNCTIONS OF 30 VARIABLES WITH 300,000 FES. "MEAN ERROR" AND "STD DEV" INDICATE THE AVERAGE AND STANDARD DEVIATION OF THE FUNCTION ERROR VALUES OBTAINED IN 25 RUNS, RESPECTIVELY. WILCOXON'S RANK SUM TEST AT A 0.05 SIGNIFICANCE LEVEL IS PERFORMED BETWEEN THE ADAPTIVE CODE AND CODE.

Function		Adaptive CoDE Mean Error±Std Dev	CoDE Mean Error±Std Dev
Unimodal Functions	F_1	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	F_2	4.65E-16±1.14E-15＋	1.69E-15±3.95E-15
	F_3	1.01E+05±5.91E+04≈	1.05E+05±6.25E+04
	F_4	6.61E-03±1.45E-02≈	5.81E-03±1.38E-02
	F_5	4.87E+02±4.31E+02－	3.31E+02±3.44E+02
Basic Multimodal Functions	F_6	1.60E-01±7.85E-01≈	1.60E-01±7.85E-01
	F_7	8.10E-03±9.19E-03≈	7.46E-03±8.55E-03
	F_8	2.01E+01±1.45E-01≈	2.01E+01±1.41E-01
	F_9	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	F_{10}	4.31E+01±1.41E+01≈	4.15E+01±1.16E+01
	F_{11}	1.15E+01±3.00E+00≈	1.18E+01±3.40E+00
	F_{12}	3.07E+03±5.05E+03≈	3.05E+03±3.80E+03
Expanded Multimodal Functions	F_{13}	1.49E+00±3.32E-01≈	1.57E+00±3.27E-01
	F_{14}	1.23E+01±5.53E-01≈	1.23E+01±4.81E-01
Hybrid Composition Functions	F_{15}	3.97E+02±6.73E+01≈	3.88E+02±6.85E+01
	F_{16}	7.48E+01±4.97E+01≈	7.37E+01±5.13E+01
	F_{17}	7.03E+01±4.04E+01≈	6.67E+01±2.12E+01
	F_{18}	9.04E+02±1.05E+00≈	9.04E+02±1.04E+00
	F_{19}	9.05E+02±1.11E+00－	9.04E+02±9.42E-01
	F_{20}	9.05E+02±9.69E-01－	9.04E+02±9.01E-01
	F_{21}	5.00E+02±4.70E-13≈	5.00E+02±4.88E-13
	F_{22}	8.61E+02±2.09E+01≈	8.63E+02±2.43E+01
	F_{23}	5.34E+02±4.18E-04≈	5.34E+02±4.12E-04
	F_{24}	2.00E+02±2.85E-14≈	2.00E+02±2.85E-14
	F_{25}	2.11E+02±7.94E-01≈	2.11E+02±9.02E-01
－		3	
＋		1	
≈		21	

"－", "＋", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoDE, respectively.

mechanisms have been proposed for adjusting the control parameters ([4], [25]). A question which arises naturally is whether CoDE can be improved by adaptive control parameter selection mechanism.

To study this question, we introduced the adaptive mechanism proposed in [4] to CoDE and implemented an adaptive CoDE. For each trial vector generation strategy at generation G, the adaptive CoDE records:

- $n_{k,G}$: the number of the trial vectors generated by this strategy with control parameter setting k (k=1, 2, and 3).
- $ns_{k,G}$: the number of the trial vectors generated by this strategy with control parameter setting k (k=1, 2, and 3) which can enter the next generation.

It also needs an additional parameter, LP, which is called the learning period. During its first LP generations, each trial vector generation strategy chooses the three control parameter settings with the same probability (i.e., 1/3). When the generation number G is larger than LP, the probability, $p_{k,G}$, of using control parameter setting k is calculated as follows:

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} n_{k,g}} + \varsigma \qquad (10)$$

and

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{3} S_{k,G}} \qquad (11)$$

where k=1, 2, and 3, $G>LP$, the first term in the right hand side of equation (10) is the success rate of control parameter setting k during the previous LP generations, and ς is set to 0.01 in our experiments to prevent $S_{k,G}$ from becoming zero. We use the roulette wheel selection to select one control parameter setting based on equation (11). Clearly, the larger $S_{k,G}$, the larger the probability $p_{k,G}$.

Following the suggestion in [4], LP is set to 50 in our experimental studies. Except adaptively choosing the control parameter settings, all the other parts of the adaptive CoDE are the same as in the original version of CoDE.

Table IV indicates that the adaptive CoDE outperforms CoDE on one unimodal function and CoDE wins the adaptive CoDE also on another unimodal function. The adaptive CoDE performs similarly with CoDE on basic multimodal functions and expanded multimodal functions. On hybrid composition functions, two methods perform similarly on 9 out of 11 test functions and CoDE wins on 2 such functions.

From the last three rows of Table IV, we can conclude that the overall performance of CoDE is slightly better than that of the adaptive CoDE. It implies that the direct use of the mechanism from [4] in CoDE cannot improve the performance of CoDE very much. Therefore, much effort is needed to study how to adapt the control parameter settings of CoDE in an adaptive manner.

VI. CONCLUSION

Many experiences in using different trial vector generation strategies and the DE control parameter settings have been reported in the literature. A comprehensive use of these experiences should be an effective way for improving the DE performance. CoDE, proposed in this paper, represented one of the first attempts along this direction. It employed three trial vector generation strategies and three control parameter settings. These strategies and parameter settings have distinct advantages and therefore they can complement one another. In CoDE, each strategy generated its trial vector with a parameter setting randomly selected from the parameter candidate pool. The structure of CoDE is simple and it is easy to implement. Moreover, under our framework, a user can easily build his/her own strategy candidate pool and parameter candidate pool for solving his/her different problems.

The experimental studies in this paper were carried out on 25 global numerical optimization problems used in the CEC2005 special session on real-parameter optimization. CoDE was compared with four other state-of-the-art DE variants, i.e., JADE, jDE, SaDE, and EPSDE, and three non-DE variants, i.e., CLPSO, CMA-ES, and GL-25. The experimental results suggested that its overall performance was better than the seven competitors. In addition, the effectiveness of random selection of control parameter settings for the trial vector generation strategies was experimentally studied.

In the future, we will generalize our work to other EAs for other hard optimization problems. For example, it is very interesting to study how to combine several different well-studied EAs in our framework such that the resultant algorithms can effectively complement one another. It is also worthwhile building a knowledge database for storing the experiences and properties of the existing algorithms.

The MATLAB source codes of the proposed CoDE, the adaptive CoDE, and other seven methods (i.e., JADE, jDE, SaDE, EPSDE, CLPSO, CMA-ES, and GL-25) can be downloaded from Qingfu Zhang's homepage: http://dces.essex.ac.uk/staff/ qzhang/

ACKNOWLEDGEMENT

The authors would like to thank Dr. J. Zhang, Dr. J. Brest, Dr. R. Mallipeddi, Dr. N. Hansen, and Dr. C. Garcia-Martinez for providing the source codes of JADE, jDE, EPSDE, CMA-ES, and GL-25, respectively, and Dr. P. N. Suganthan for providing the source codes of SaDE and CLPSO. The authors also thank the guest editors and the three anonymous reviewers for their very constructive comments and suggestions. The first author appreciates Miss Y. Yang's inspiration.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep. TR-95-012, 1995.

[2] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Opt.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[3] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.

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

[5] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evolut. Comput.*, vol. 13, no. 3, pp. 526-553, 2009.

[6] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evolut. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[7] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimization with differential evolution," *in Proc. IEEE Congr. Evol. Comput. (CEC-2005)*, vol. 1. Piscataway, NJ: IEEE Press, pp. 506–513.

[8] H. Y. Fan and J. Lampinen, "A trigonometric mutation operator to differential evolution," *J. Global Optim.*, vol. 27, no. 1, pp. 105–129, 2003.

[9] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello. Coello, "Modified differential evolution for constrained optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC' 2006)*, IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 332-339.

[10] V. Feoktistov, S. Janaqi, "Generalization of the strategies in differential evolution," in *Proc. of the 18th Int Parallel and Distributed Processing Symposium*. Santa Fe, 2004, pp. 165-170.

[11] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genetic Evol. Comput. Conf. (GECCO 2006)*, pp. 485–492.

[12] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evolut. Comput.*, vol. 13, no. 5, pp. 945-958, 2009.

[13] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *in Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, A. Grmela and N. E. Mastorakis, Eds. Interlaken, Switzerland: WSEAS Press, 2002, pp. 293–298.

[14] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, Jun. 2005, pp. 991–998.

[15] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 6, pp. 448–462, 2005.

[16] 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. Evolut. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[17] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 637–686, 2006.

[18] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, in press, 2010.

[19] K. V. Price, "An introduction to differential evolution," *in New Ideas Optimization*. London, U.K.: McGraw-Hill, 1999, pp. 293–298.

[20] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, May 2005, IIT Kanpur, India.

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

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

[23] C. Garcia-Martinez, M. Lozano, F. Herrera, D. Molina, and A.M. Sanchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, pp. 1088–1113, 2008.

[24] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. of the IEEE Congress on Evolutionary Computation, (CEC-2005)*, 2005, pp.1769-1776.

[25] J. Tvrdík, I. Křivý, L. Mišík, "Adaptive population-based search: application to estimation of nonlinear regression parameters," *Computational Statistics and Data Analysis*, vol. 52, no. 2, pp. 713-724, 2007.

[26] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," *Swarm Evolutionary and Memetic Computing Conference*, Chennai, India, 2010.