Differential Evolution With an Individual-Dependent Mechanism

Lixin Tang, *Senior Member, IEEE,* Yun Dong, and Jiyin Liu

***Abstract*—Differential evolution (DE) is a well-known optimization algorithm that utilizes the difference of positions between individuals to perturb base vectors and thus generate new mutant individuals. However, the difference between the fitness values of individuals, which may be helpful to improve the performance of the algorithm, has not been used to tune parameters and choose mutation strategies. In this paper, we propose a novel variant of DE with an individual-dependent mechanism that includes an individual-dependent parameter (IDP) setting and an individual-dependent mutation (IDM) strategy. In the IDP setting, control parameters are set for individuals according to the differences in their fitness values. In the IDM strategy, four mutation operators with different searching characteristics are assigned to the superior and inferior individuals, respectively, at different stages of the evolution process. The performance of the proposed algorithm is then extensively evaluated on a suite of the 28 latest benchmark functions developed for the 2013 Congress on Evolutionary Computation special session. Experimental results demonstrate the algorithm's outstanding performance.**

***Index Terms*—Differential evolution (DE), global numerical optimization, individual dependent, mutation strategy, parameter setting.**

I. Introduction

DIFFERENTIAL evolution (DE), proposed by Storn and Price [1] in 1995, is an efficient population-based global searching algorithm for solving optimization problems [2]–[4]. Over the years, different variants of DE have been developed to solve complicated optimization problems in a wide range of application fields, such as auction [5], decision making [6], neural network training [7], chemical engineering [8], robot control [9], data clustering [10], gene regulatory network [11], aircraft control [12], and scheduling in steel production [13].

Manuscript received October 20, 2013; revised February 25, 2014 and July 7, 2014; accepted August 30, 2014. Date of publication September 30, 2014; date of current version July 28, 2015. This work was supported in part by the State Key Program of National Natural Science Foundation of China under Grant 71032004, in part by the Fund for Innovative Research Groups of the National Natural Science Foundation of China under Grant 71321001, and in part by the Fund for the National Natural Science Foundation of China under Grant 61374203.

L. Tang is with Liaoning Key Laboratory of Manufacturing System and Logistics, Institute of Industrial Engineering and Logistics Optimization, Northeastern University, Shenyang 110819, China (e-mail: lixintang@mail.neu.edu.cn).

Y. Dong is with State Key Laboratory of Synthetical Automation for Process Industry, Institute of Industrial Engineering and Logistics Optimization, Northeastern University, Shenyang 110819, China (e-mail: dydexter@hotmail.com).

J. Liu is with the School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, U.K. (e-mail: j.y.liu@lboro.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2014.2360890

DE is a typical evolutionary algorithm (EA), which employs strategies inspired by biological evolution to evolve a population. Throughout this paper we suppose that DE is used for solving minimization problems, and that the objective function can be expressed as $f(\mathbf{x})$, $\mathbf{x} = (x^1, \ldots, x^D) \in R^D$. We take the objective function value of each solution as its fitness value in DE. Note that with this setting a solution with a lower fitness value is a better solution. Some previous studies (e.g., [6] and [7]) also used such settings, while [2] and [3] used objective function only without defining fitness. To solve the problem, the classic DE process starts from an initial population of NP individuals (vectors) in the solution space, with each individual representing a feasible solution to the problem. At each generation g, the individuals in the current population are selected as parents to undergo mutation and crossover to generate offspring (trial) individuals. Each individual $\mathbf{x}_{i,g}$ $(i = 1, 2, \ldots, \mathrm{NP})$ in the population of current generation is chosen once to be a parent (target individual) for crossover with a mutant individual $\mathbf{v}_{i,g}$ obtained from a mutation operation in which the base individual is perturbed by a scaled difference of several randomly selected individuals. The offspring individual $\mathbf{u}_{i,g}$ generated from the crossover contains genetic information obtained from both the mutant individual and the parent individual. Each element $u^j{}_{i,g}$ $(j = 1, 2, \ldots, D)$ of $\mathbf{u}_{i,g}$ should be restricted within the corresponding upper and lower boundaries. Otherwise, it will be reinitialized within the solution space. The mutation and crossover operations of classic DE are illustrated in Fig. 1. The fitness value (objective function value) of the offspring individual $\mathbf{u}_{i,g}$ is then compared with the fitness value of the corresponding parent individual $\mathbf{x}_{i,g}$, and the superior individual is chosen to enter the next generation. Then, the new population is taken as the current population for further evolution operations. This continues until specific termination conditions are satisfied. In the final generation, the best individual will be taken as the solution to the problem.

There are only a few DE control parameters that need to be set: the population size NP, the mutation factor (scale factor) F, and the crossover probability crossover rate (CR). Different parameter settings have different characteristics. In addition, the strategies used for each operation in DE, especially the mutation strategies, can vary to obtain diverse searching features that fit different problems. Therefore, the ability of DE to solve a specific problem depends heavily on the choice of strategies [4], and the setting of control parameters [14], [15]. Inappropriate configurations of mutation strategies and control parameters can cause stagnation

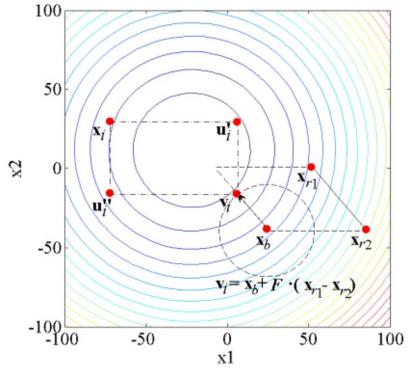

Fig. 1. 2-D plot of evolution operation of classic DE.

due to over exploration, or can cause premature convergence due to over exploitation. Exploration can make the algorithm search every promising solution area with good diversity, while exploitation can make the algorithm execute a local search in some promising solution areas to find the optimal point with a high convergence rate. Therefore, choosing strategies and setting parameters aimed at getting a good balance between the algorithm's effectiveness (solution quality) and efficiency (convergence rate) have always been subjects of research.

In the evolution process of a DE population, superior individuals and inferior individuals play different roles. The former guide the searching direction, and the latter maintain population diversity. Generally, in the solution space, better solutions are more likely to be found in the neighborhoods of superior individuals or in areas relatively far from inferior individuals. Unfortunately, these differences have not been reflected in the control parameter setting and mutation strategy selection in classic DE and its variants.

In this paper, a novel DE variant (IDE) with an individual-dependent mechanism is presented. The new mechanism in IDE includes an individual-dependent parameter (IDP) setting and an individual-dependent mutation (IDM) strategy. The IDP setting first ranks individuals based on their fitness values and then determines the parameter values of the mutation and crossover operations for each individual based on its rank value. In IDP, superior individuals tend to be assigned with smaller parameter values so as to exploit their neighborhoods in which better solutions are likely contained, while inferior individuals tend to be assigned with larger parameter values so as to explore further areas in the solution space. The IDM strategy assigns four mutation operators with different searching characteristics for superior and inferior individuals at different stages of the iteration process. Further, the IDM strategy introduces perturbations to the difference vector by employing diverse elements generated from the solution space. In response to the decreased diversity of the population, we propose to increase the proportions of the explorative mutation operator and the perturbations following an exponential function. Extensive experiments are carried out to evaluate the performance of IDE by comparing it with five IDE variants, five state-of-the-art DE variants, ten up-to-date DE variants, and three other well-known EAs on the latest 28 standard benchmark functions listed in the CEC 2013 contest.

The remainder of this paper is organized as follows. In Section II, the related works on DE mutation strategy selecting and control parameter setting are reviewed. After the development of the IDP setting and the IDM strategy, the complete procedure of IDE is presented in Section III. In Section IV, experiment results are reported to demonstrate the effectiveness of IDE. Section V gives the conclusion and indicates possible future work.

II. Literature Review

Over the past decades, researchers have developed many different mutation strategies [16], [17]. It has been shown that some of them are fit for global search with good exploration ability and others are good at local search with a good exploitation ability [3]. For example, mutation strategies with two difference vectors (e.g., DE/rand/2) can increase the population diversity compared to strategies with a single difference vector (e.g., classic DE/rand/1), while mutation strategies taking the best individual of the current generation in the base vector (e.g., DE/best/1 and DE/current-to-best/1) can strengthen exploitive search and so speed up convergence.

However, to improve algorithm robustness, exploration, and exploitation must be simultaneously combined into the evolution strategy. Therefore, the single-mutation operator strategies with composite searching features (such as BDE [18], DE with global and local neighborhoods (DEGL) [19], JADE [20], proposed improved DE (PIDE) [13], enhancing differential evolution utilizing proximity-based mutation operators (ProDE) [21], differential evolution with best of random mutation strategy (BoRDE) [22], and differential evolution with a trigonometric mutation operation (TDE) [23]) and the multimutation operators strategies with different searching features (such as SaDE [24], [25], teaching and learning based self-adaptive DE (TLBSaDE) [26], differential evolution with composite trial vector generation strategies and control parameters (CoDE) [27], differential evolution with ensemble of parameters and mutation strategies (EPSDE) [28], [29], super-fit multicriteria adaptive differential evolution (SMADE) [30], jDE*soo* [31], and structured population size reduction differential evolution with multiple mutation strategies (SPSRDEMMS) [32]) were proposed. Epitropakis *et al.* [18] linearly combined an explorative and an exploitive mutation operator to form a hybrid approach (BDE) with an attempt to balance these two operators. DEGL, proposed by Das *et al.* [19], combines global and local mutant individuals to form the actual mutant individuals with a weight coefficient in each generation. In JADE, proposed by Zhang and Sanderson [20], an optional external archive is combined with a mutation strategy DE/current-to-*p*best/1 that utilizes historical information to direct population searching. Tang *et al.* [13] proposed PIDE which combines three different random difference vectors into mutation operator DE/current-to-*p*best/1 to encourage diversity. The individuals in the neighborhood of the target individual are selected to participate in the mutation operation in ProDE, proposed by Epitropakis *et al.* [21]. To obtain both good diversity and fast convergence rate, Lin *et al.* [22] proposed a DE

variant with a best of random mutation strategy (BoRDE), in which the individual with the best fitness value among several randomly chosen individuals is selected as the base vector in the mutation operation. Fan and Lampinen [23] introduced a trigonometric mutation operation to form TDE (trigonometric mutation DE), in which the searching direction of each new mutant individual is biased to the best among three different individuals randomly chosen in the mutation. Each of the four given effective trial vector generation strategies (mutation and crossover operations) is selected to be implemented according to a self-adaptive probability in SaDE, proposed by Qin *et al.* [25]. In addition to a mutation strategy pool, a teaching and learning mechanism is employed by TLBSaDE, i.e., a new variant of SaDE, to generate the mutant individuals [26]. Wang *et al.* [27] proposed CoDE in which three mutation strategies are implemented in parallel to create three new mutant individuals for crossover with the target individual before selecting one to enter the next generation. Mallipeddi *et al.* [28] proposed EPSDE, in which a set of mutation strategies along with a set of parameter values compete to generate offspring. Zhao and Suganthan [29] modified EPSDE by incorporating an SaDE type learning scheme. SMADE, proposed in [30], makes use of a multiple mutation strategy consisting of four different mutation operators and selects one operator via a roulette-wheel selection scheme for each target individual in current generation. Brest *et al.* [31] proposed a variant of jDE, i.e., jDE*soo* (jDE for single objective optimization), which concurrently applies three different DE strategies in three sub-populations. In SPSRDEMMS [32], one mutation operator is selected from DE/rand/1 and DE/best/1 to generate the mutant individual according to the population size gradually reduced with iteration progresses. A ranking-based mutation operator is introduced in Rank-DE [33] where the base and terminal (guiding) individuals in mutation operators are proportionally selected according to their ranking values in current generation.

With respect to parameter setting methods, we can classify them into three categories: constant, random, and adaptive (including self-adaptive). In constant parameter setting, such as that used in classic DE, parameters are preset before the search and kept invariant during the entire iteration process. Storn and Price [3] stated that it is not difficult to choose control parameters for obtaining good results. In their opinion, a promising range of NP is between 5D and 10D, where D is the dimension of the individual, 0.5 represents a reasonable value of F, and 0.1 is a first choice for CR for general situations and 0.9 for situations in which quick solutions are desired. However, based on the results of tests on four benchmark functions, Gämperle *et al.* [15] concluded that DE performance depends heavily on the setting of control parameters. The researchers stated that a promising range of NP is between 3-D and 8D, an efficient initial value for F is 0.6, and a promising range for CR is [0.3, 0.9]. Rönkkönen *et al.* [34] suggested that a reasonable range of NP is between 2-D and 40D, that F should be sampled between 0.4 and 0.95 (and that 0.9 offered a compromise between exploration and exploitation), and that CR should be drawn from the range (0.0, 0.2) if the problem is separable, or [0.9, 1] if the problem is both nonseparable and multimodal. They set $F = 0.9$, CR = 0.9, and NP = 30 for all experimental functions in the CEC 2005 contest benchmark suite. In CoDE [27], each trial vector generation strategy randomly selects one parameter setting from a pool consisting of three constant parameter settings. Differential evolution with a orthogonal crossover, proposed in [35], employs a new orthogonal crossover operator to improve the searching ability of classic DE with $F = 0.9$, CR = 0.9, and NP = D. In [36], differential evolution with automatic parameter configuration (DE-APC) randomly assigns F value and CR value from two preset constant sets F_{set} and CR_{set}, respectively, to each individual. The diversity of the conclusions reached by these researchers indicates the number of different claims on the DE control parameter setting. It is impossible that one constant parameter setting fits all problems, instead, effective constant parameters should be problem-dependent.

To avoid manually tuning parameters, researchers have developed some techniques to automatically set the parameter values, one of which is random parameter setting. Linear variation, probability distribution, and specified heuristic rules are usually employed to generate the diverse parameters in random parameter setting. For example, Das *et al.* [37] presented two ways to set F in DE, i.e., random and time varying. In the random way, F is set to be a random real number from the range (0.5, 1); in the time varying way, F is linearly reduced within a given interval. In SaDE [25], the value of parameter F is randomly drawn from a normal distribution $N(0.5, 0.3)$ for each target individual in the current population. Abbass [38] generated F from the standard normal distribution $N(0, 1)$. CR in [39] is generated from a normal distribution $N(0.5, 0.15)$. CR is set to be $[2^{1/(n\alpha_e)}]^{-1}$ in differential evolution with concurrent fitness based local search (DEcfbLS) [40], where n is the dimension of the problem and α_e is the inheritance factor [41]. The random setting can improve the exploration ability by increasing searching diversity.

Another automatic parameter setting method focuses on the adaptation of control parameters. In adaptive parameter setting, the control parameters are adjusted according to the feedback from the searching process [42], [43], or undergo evolutionary operation [38], [44]. Incorporating the individuals of successive generations and relative objective function values as inputs, Liu and Lampinen [42] employed a fuzzy logic controller to tune the parameters adaptively in fuzzy adaptive DE (FADE). A self-adaptive DE (jDE) proposed by Brest *et al.* [43] assigns the values from the ranges [0.1, 1.0] and [0.0, 1.0] in an adaptive manner with probabilities τ_1 and τ_2 to F and CR, respectively, for each mutation and crossover operation. In JADE [20], according to the historical successful information of parameters, F is generated by a Cauchy distribution, and CR is sampled from a normal distribution for each individual at each generation. SHADE [45] is an improved version of JADE which employs a different success-history based mechanism to update F and CR. SaDE [25] adaptively adjusts CR values following a normal distribution with the mean value depending on the previous successful CR values. The parameters in [29] are gradually self-adapted by learning from the success

history. In [46], for each individual, control parameters is selected adaptively from a set of 12 different DE parameter settings with a probability depending on the corresponding success rate. Instead of the single one value of F to one individual, population's variance-based adaptive differential evolution (PVADE), presented in [47], adopts a scaling factor vector F_m, calculated by a diversity measure of the population in each dimension level. The mutation and crossover rates of the EAs [38], [44], proposed based on DE, are self-adapted by the DE mutation operation.

We have observed that the existing mutation strategies and control parameter setting methods reviewed above are used at the population level, i.e., all individuals in the current population are treated identically without consideration of the differences in their fitness values. Consideration of this difference may be helpful for finding better solutions in the next generation. We therefore propose a new difference-based mechanism in which the control parameters and mutation operators are set to each individual according the information on its fitness value in the current generation to improve the DE performance in both convergence rate and diversity.

III. Proposed IDE

In essence, the purposes of DE mutation operations and control parameters are to determine the searching direction and the size of the searching area, respectively. In this section, we first discuss the influences of the control parameters on DE and give the IDP setting method. Then, we investigate the convergence features of DE with different mutation operators and propose the IDM strategy. Finally, we present the complete IDE procedure with these new strategies.

Specifically, to demonstrate the validity of the proposed strategies in this section, four typical minimization problems introduced in the CEC 2013 benchmark functions suite [48] are employed to run the pilot experiments. The contour maps and the 3-D maps for 2-D benchmark functions are plotted in Fig. 2. The sphere function [see Fig. 2(a)] is unimodal, continuous, differentiable, and separable. It is the benchmark for studying the convergence features. The rotated Schwefel's function [see Fig. 2(b)] is multimodal, rotated, nonseparable, and asymmetrical. Its number of local optima is huge, and its second best local optimum is far from the global optimum [48]. The function is very difficult to optimize and is a good candidate for studying the ability to jump out of local optima. The rotated Schaffers F7 function [see Fig. 2(c)] is multimodal, nonseparable, asymmetrical, and has a huge number of local optima. It is very suitable for investigating the exploration ability. The rotated expanded Scaffer's F6 function [see Fig. 2(d)] is multimodal, nonseparable, and asymmetrical. Through it, the exploitation ability can be examined.

A. IDP Setting

From the illustration of the evolution operation of classic DE in Fig. 1, we can observe that the scale factor F is the parameter that controls the size of the searching area around the base individual $\mathbf{x}_b$, and the CR indicates the probability of inheriting elements from the mutant individual $\mathbf{v}_i$ in the process of constructing the trial individual $\mathbf{u}_i$.

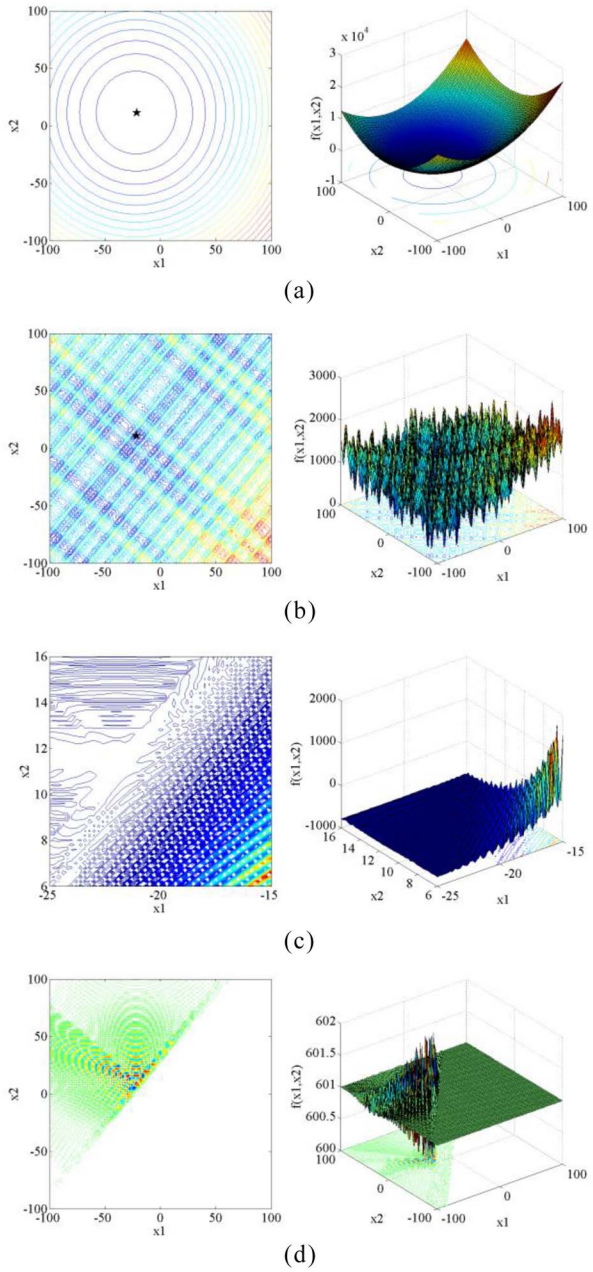

Fig. 2. Contour maps and the 3-D maps of four benchmark functions for the pilot experiments. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

We look at the effects of the control parameters in two cases, i.e., the unimodal sphere function and the multimodal rotated Schwefel's function. For the sphere function expressed in Fig. 2(a), it can be seen from the diagram that the individuals with better (smaller) fitness values have shorter distances to the global optimum in the center (marked by a black star) of concentric circles, whereas individuals with worse (larger) fitness values have longer distances. This suggests that to find better solutions, for superior base individuals, the mutation operations should exploit the searching area with a relatively small radius by employing a relatively small F value, and for inferior base individuals, they should jump out of the neighborhood area to explore more promising areas by using a relatively large value of scale factor. In terms of crossover,

the offspring individuals should inherit more elements from the corresponding parent (target) individuals with a smaller CR value when the parent individuals are superior. In contrast, if the parent individuals are far away from the optimum, more promising offspring individuals are likely to be obtained by accepting more mutant elements with a larger CR value.

Based on this analysis, we propose to set the parameters F and CR in accordance with the differences between individuals' fitness values. In other words, we propose that parameter setting should follow a general principle that the parameters for individuals with higher fitness values are larger, and vice versa. Using this principle, specific IDP setting can be devised. Two natural schemes are given below.

1) Rank-Based Scheme: Reindex all individuals in the current population in ascending order of their fitness (objective function) values, i.e., individual $\mathbf{x}_i$ is the ith most superior one. The control parameter F_b associated with base individual $\mathbf{x}_b$ and CR_i associated with target individual $\mathbf{x}_i$ can be set using (1) and (2), respectively, as follows:

$$F_b = \frac{b}{\mathrm{NP}} \tag{1}$$

$$\mathrm{CR}_i = \frac{i}{\mathrm{NP}}. \tag{2}$$

2) Value-Based Scheme: Denote f_i as the fitness value of individual $\mathbf{x}_i$ in the current population. F_b and CR_i can be set as

$$F_b = \frac{f_b - f_L + \delta_L}{f_U - f_L + \delta_L} \tag{3}$$

$$\mathrm{CR}_i = \frac{f_i - f_L + \delta_L}{f_U - f_L + \delta_L} \tag{4}$$

where f_L and f_U are the minimum and maximum fitness values, respectively, in the current population, and δ_L is the absolute value of the difference between the minimum fitness value and the second minimum (different from the minimum value) in the searching history up to the current population. Note that there may be more than one solution with the minimum fitness value, and the second minimum value has to be different from this minimum. The inclusion of δ_L prevents the parameters to be set to 0. Note also that it is not necessary to use the same scheme to set F and CR.

The situation for multimodal functions is more complicated than that for unimodal functions. We take as an example a typical multimodal case in Fig. 2(b), i.e., the rotated Schwefel's function. As we can observe, superior individuals are not always close to the global minimum (marked by a black star) because many local minima spread all over the searching space and confuse the search. In fact, relative to the superior individuals, the inferior individuals may be closer to the global minimum. Based on this observation, we randomize the parameters using a normal distribution with the mean specified to the original value and the standard deviation specified to 0.1. Denote a random number from a normal distribution by *randn*(*mean*, *std*) [20]. Then, the IDP setting can be modified as

$$F_b^{'} = randn(F_b, 0.1) \tag{5}$$

$$\mathrm{CR}_i^{'} = randn(\mathrm{CR}_i, 0.1). \tag{6}$$

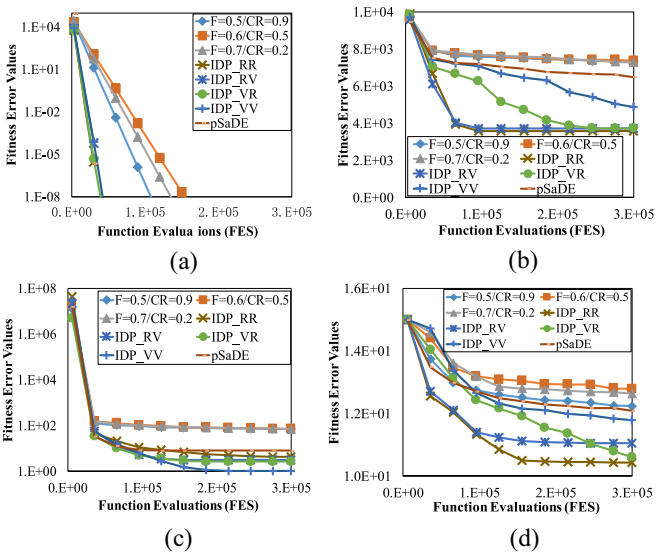

Fig. 3. Convergence curves of DEs with different parameter settings on four 30-D benchmark functions. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

With the randomization, the control parameter values near their mean values maintain search efficiency, while the parameter values away from those improve search diversity. If the parameter value drawn from the normal distribution is beyond the range of (0, 1), it will be ignored and a new value will be drawn until the parameter value is within the range.

To verify the idea of IDP setting, we run a pilot experiment on the four benchmark functions introduced above. The dimension of the problems (D) and the population size (NP) are set to 30 and 100, respectively. Based on the strategy DE/rand/1/bin, DEs with different parameter settings are applied to optimize the benchmark functions. These include [$F = 0.5/\mathrm{CR} = 0.9$] [3], [$F = 0.6/\mathrm{CR} = 0.5$] [15], [$F = 0.7/\mathrm{CR} = 0.2$] [34], the parameter setting in SaDE [25] (denoted by parameter setting in SaDE), and the IDPs presented above. Four versions of IDP are tested. We use two letters to indicate the IDP parameter setting schemes, the first for F and the second for CR. For example, IDP_RV means that F is set using the rank-based scheme and CR is set using the value-based scheme. For all pilot experiments, the maximum function evaluations (MaxFES) is set to 300 000 = 10 000D (i.e., the iteration number $g_{\max}$ is 3000 = MaxFES/NP) and the fitness error value of the best individual is sampled every 30 000 FES. For convenient illustration, the convergence curves of different DEs on four problems are plotted in Fig. 3. The horizontal axis is the number of FES, and the vertical axis is the average of the fitness error values over 51 [48] independent runs.

The parameters generated by the value-based scheme adjust the size of the searching area more precisely, and the parameters generated by the rank-based scheme are uniformly distributed in the range [0, 1]. The diagrams clearly show that the IDP settings outperform other parameter settings in all cases. Among the four IDPs, IDP_RR achieves the best performance and is selected as the parameter setting in IDE.

A recently proposed DE variant Rank-DE [33] employs a similar ranking-based scheme to select some individuals in mutation operators. This is different from IDE, which uses the scheme to calculate the values of the control parameters. The comparison of these two algorithms is carried out in Section IV.

B. IDM Strategy

Mutation is an operation for obtaining the mutant individuals in the area around the base individual. In the mutation operator of classic DE, the base individual serves as the center point of the searching area, the difference vector sets the searching direction, and the scale factor controls the movement step. Considering the mutation and crossover strategies used, the evolutionary operation of DE can be denoted using a four-parameter notation DE/base/differ/cross [3]. There are several widely used mutation strategies with different kinds of base vectors and different numbers of difference vectors, e.g., DE/rand/1, DE/best/1, DE/current-to-best/1, DE/rand/2, and DE/best/2. The single base vector mutation strategy DE/best/differ performs better in convergence rate, since the mutation operation is implemented as a local search around the best individual. However, it is very easy to be trapped in local minima, leading to premature convergence. On the contrary, the single base vector mutation strategy DE/current/differ, which uses the target (current) individual as the base vector, has a high diversity and performs better in global searching ability. In this case, without the best individual guiding the evolution, the algorithm searches the solution space sufficiently but slowly and aimlessly. Therefore, the strategy DE/rand/differ, in which the base vector is randomly selected from the current population at each generation, can be considered as a balance between DE/best/differ and DE/current/differ. As for the operator DE/current-to-best/1, it can be employed as a mixed strategy that combines a current individual and a best individual as the origin (start) individual and the guiding (terminal) individual, respectively, to form a composite base vector. To clearly define the mutation operators with composite base vectors in this paper, we denote them as DE/origin-to-guiding/differ, where "origin" and "guiding" are to be specified.

To illustrate the searching characteristics of these four mutation strategies, i.e., DE/rand/1, DE/best/1, DE/current/1, and DE/current-to-best/1, we use the same four benchmark functions and experiments configuration above, and we set $F = 0.5$ and CR = 0.5. The convergence curves are plotted in Fig. 4.

The experiment results demonstrate that, among the four, DE/best/1 is the fastest strategy in unimodal function, but also the most unstable strategy in multimodal functions due to premature convergence. While a good diversity enables DE/current/1 to avoid premature convergence, its convergence rate is poor. DE/current-to-best/1 does not achieve the best performance in convergence rate or in population diversity, but it can locate the global optimum faster than DE/current/1, and it can escape from local minima with better diversity than DE/best/1. DE/rand/1 is a robust strategy with strong performance in the multimodal functions, especially at the later

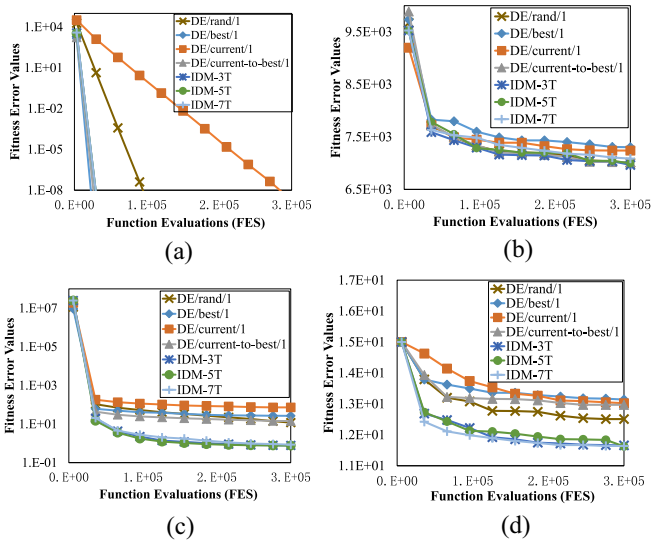

Fig. 4. Convergence curves of DEs with different mutation strategies on four 30-D benchmark functions. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

iteration process. These results suggest that a better performance may be obtained by using a composite base vector that integrates a diversity element and a rapidity element, and by incorporating a random individual in the mutation operation.

In addition, from the convergence curves plotted in Figs. 3 and 4, we can observe that most algorithms perform more effectively with a high convergence rate in the earlier stage of the searching process relative to the later stage, especially for multimodal functions. At the later stage, the algorithms are easily trapped in local optimal solutions due to poor population diversity. To investigate the variation of the diversity of the population, we apply the classical DE (DE/rand/1/bin with $F = 0.5$ and CR = 0.5) to optimize the four benchmark functions and observe the diversities of individuals and fitness values in each generation. We define the diversity indicators for each generation as the standard deviations of the individuals and of their fitness values. They are denoted as diversity of individuals (DI) and diversity of fitness (DF), respectively, and can be calculated as

$$\mathrm{DI} = \frac{1}{\mathrm{NP}} \sqrt{\sum_{i=1}^{\mathrm{NP}} \left\| \mathbf{x}_i - \frac{1}{\mathrm{NP}} \sum_{j=1}^{\mathrm{NP}} \left(\mathbf{x}_j\right) \right\|^2} \tag{7}$$

$$\mathrm{DF} = \frac{1}{\mathrm{NP}} \sqrt{\sum_{i=1}^{\mathrm{NP}} \left(f\left(\mathbf{x}_i\right) - \frac{1}{\mathrm{NP}} \sum_{j=1}^{\mathrm{NP}} f\left(\mathbf{x}_j\right) \right)^2}. \tag{8}$$

The curves of diversity indicators DI and DF are plotted in Fig. 5. The curves show that DI and DF decrease gradually as the iteration continues. In other words, the diversity of the population becomes poorer with the algorithm iteration. This is the inevitable trend of algorithm convergence. However, if the algorithm has not yet focused on the neighborhood of the global optimal solution at the later stage of the searching process, it is very hard to jump out of the local minima

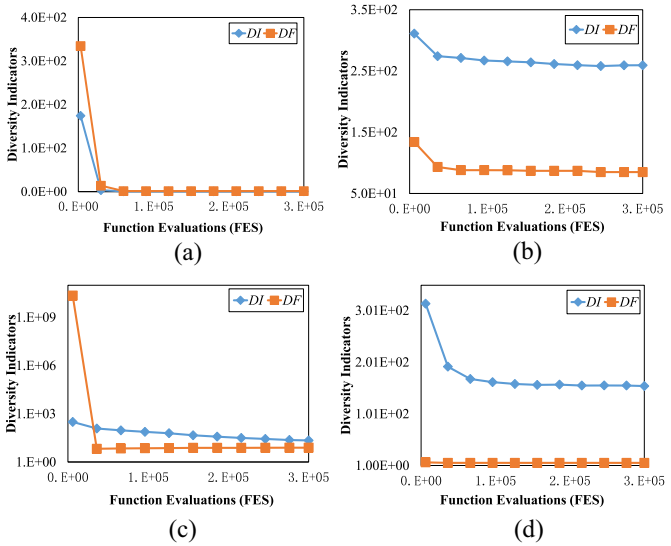

Fig. 5. Curves of the diversity of individuals (DI) and the diversity of fitness (DF) values on four 30-D benchmark problems. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

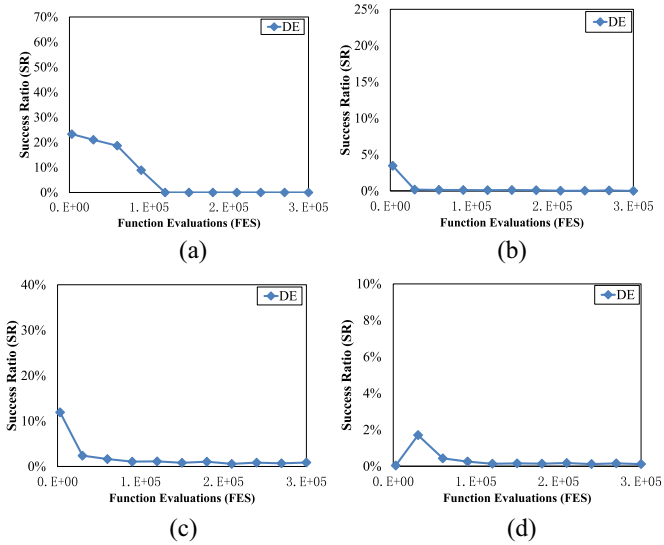

Fig. 6. Success ratio curves of classic DE on four 30-D benchmark problems. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

without good exploration ability. Meanwhile, the difference between the superior individuals and the inferior individuals are insignificant when they gather together around the best solution found so far at the later stage.

Considering the analysis above, we first adopt the "current" individual as the origin individual (DE/current-to-guiding/1) for the purpose of diversity at the earlier stage, and we then employ the "rand" individual as the origin individual (DE/rand-to-guiding/1) to accelerate the convergence rate at the later stage. We combine these origin individuals according to their different fitness values with other specified individuals as the guiding ones to form the composite base vectors to guide the searching direction. To utilize the fitness information of the origin individuals, we first use the reindex result in the IDP setting to classify the individuals in the population into two sets, i.e., a superior set S and an inferior set I, and then choose the guiding individuals for the origin individuals in different sets. The set S contains a proportion of individuals in the generation with the best rankings. Only with the proper direction will the searching of the population be more efficient. For individuals in S, because better solutions tend to be found in their neighborhoods, an omnidirectional local search should be implemented. We therefore employ DE/origin-to-rand/1 to generate the mutant individuals when the corresponding origin individuals are superior. For individuals in I, we employ relatively better individuals to guide their searching directions and thus present a mutation operator DE/origin-to-better/1.

This new mutation strategy can be expressed as (9). Where o is the index of the origin individual which is the current individual ($o = i$) at the earlier stage of the iteration process, and a random individual at the later stage of the process, according to the previous analysis. Indexes r_1, r_2, and r_3 are randomly selected from the range [1, NP] and are different from each other and from the target index i. *better* is the index of the better individual randomly selected from the superior individual set S in the current generation

$$\mathbf{v}_{i,g} = \mathbf{x}_{o,g} + \begin{cases} F_o \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{o,g}) + F_o \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) & o \in S \\ F_o \cdot (\mathbf{x}_{better,g} - \mathbf{x}_{o,g}) + F_o \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) & o \in I \end{cases} \quad o \neq r_1 \neq r_2 \neq r_3. \tag{9}$$

Now, the issue is how to select one generation index threshold g_t between 1 and $g_{\max}$ to divide the iteration process into two stages, i.e., the earlier and the later, to determine the specified origin individuals $\mathbf{x}_{o,g}$ in (9). From the convergence curves plotted in Figs. 3 and 4, we can observe that an algorithm has different convergence rates on different problems with different difficulty degrees. Therefore, g_t varies for different problems. To further express the convergence process, the success ratio (SR) curves of classic DE on solving four 30-D problems are plotted in Fig. 6. The success ratio of each generation g is defined by

$$\mathrm{SR}_g = \frac{\mathrm{NS}_g}{\mathrm{NP}} \tag{10}$$

where NS_g is the number of offspring individuals that can successfully enter the next generation $g+1$. Because SR tends to decrease continually from a relative larger value as the iteration progresses, it can be employed as an indicator to identify whether the earlier stage ends. Generally, we can say that an algorithm enters the later stage of the convergence process when SR_g equals to zero for a given number of consecutive generations T. For the sphere function [Fig. 6(a)] and the rotated Schwefel's function [Fig. 6(b)], SR can decrease to near zero after a number of iterations with efficient searching. However, for the rotated Schaffers F7 function [Fig. 6(c)], and rotated expanded Scaffer's F6 function [Fig. 6(d)], the value of SR at the later stage is close to but always larger than zero until the end of iteration. Therefore, an auxiliary generation index threshold G_T and a success ratio threshold SR_T are introduced into IDM to help select the generation index

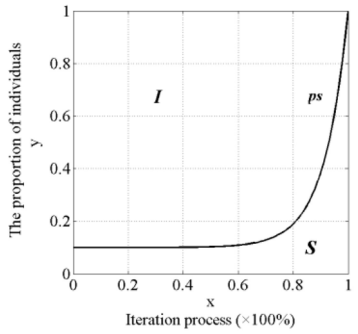

Fig. 7. Proportion model for the classification of individuals in the IDM strategy.

threshold g_t as

$$g_t = \text{Min}\left\{g' | \text{SR}_g \le \text{SR}_T, \forall g \in \left[g' - T, g'\right]\right\}$$
$$g' = T + 1, \ldots, g_{\max} \tag{11}$$

where g and $g\prime$ are the generation indexes, and T is set to 1000D/NP. In (11), SR_T is 0 before G_T+1, and SR_T is 0.1 after G_T. The value of G_T is determined in the next pilot experiment.

Further, in response to the decreased population diversity, we propose a proportion model to gradually increase the proportion of individuals in S as the algorithm progresses. The proportion of superior individuals in the population is set as the following exponential function of the generation index g:

$$ps = 0.1 + 0.9 \times 10^{5(g/g_{\max} - 1)}. \tag{12}$$

This function is plotted in Fig. 7, where the x-axis denotes the iteration process ($x = g/g_{\max}$), and the y-axis is the proportion of individuals. At any point of the iteration process, the proportion of superior individuals in the population is the height of ***ps***, and the proportion of inferior individuals is represented by the distance between ***ps*** and $y = 1$. With the proportion model, at the beginning of the iteration, most mutant individuals are generated by DE/current-to-better/1 with good diversity. And DE/current-to-rand/1 is employed to exploit the neighborhoods of the superior individuals. As the evolution progresses, an increasing number of random individuals are employed to guide the searching and maintain the diversity. Since the differences between the individual fitness values become smaller at the later stage of evolution, most individuals can be regarded as superior, and DE/rand/2 can be utilized to avoid trapping individuals into the local minima with a maximum of diversity. For the remaining, inferior individuals, DE/rand-to-better/1 is employed to accelerate the convergence.

To further enhance the global searching ability of the algorithm, we introduce perturbations in the difference vectors with small probability to help the base individuals move out of the local area. Combining the different components described above, the IDM strategy can be presented as

$$\mathbf{v}_{i,g} = \mathbf{x}_{o,g} + \begin{cases} F_o \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{o,g}\right) + F_o \cdot \left(\mathbf{x}_{r_2,g} - \mathbf{d}_{r_3,g}\right) & o \in S \\ F_o \cdot \left(\mathbf{x}_{better,g} - \mathbf{x}_{o,g}\right) + F_o \cdot \left(\mathbf{x}_{r_2,g} - \mathbf{d}_{r_3,g}\right) & o \in I \end{cases}$$
$$o \ne r_1 \ne r_2 \ne r_3. \tag{13}$$

TABLE I
MUTATION OPERATORS IN IDM STRATEGY

Generation Index	Origin Individual	Mutation Operator
$g \le g_t$	superior	DE/current-to-rand/1(p)
$g \le g_t$	inferior	DE/current-to-better/1(p)
$g > g_t$	superior	DE/rand/2(p)
$g > g_t$	inferior	DE/rand-to-better/1(p)

The indexes o, r_1, r_2, and r_3 are selected in the same way as in (9). $\mathbf{d}_{r3,g}$ is a disturbed individual derived from the current population with possible elements randomly taken from the entire solution space. Each element of $\mathbf{d}_{r3,g}$ is determined by

$$d^j_{r_3,g} = \begin{cases} L^j + rand(0,1) \cdot \left(U^j - L^j\right), \text{if} \left(rand^j_i(0,1) < p_d\right) \\ x^j_{r_3,g}, \qquad \text{otherwise} \end{cases} \tag{14}$$

where the probability factor of disturbance p_d is set to be $0.1 \times$ ***ps***, L^j and U^j are the lower and the upper bound values of the jth element. To conclude IDM, we list the mutation operators for different individuals at different stages of iteration in Table I. The notation "(p)" denotes the perturbations introduced to difference vectors. It can be summarized that, in the new strategy, the origin individuals are current and random individuals at earlier ($g \le g_t$) and later stages ($g > g_t$), respectively. Additionally, the guiding individuals are random individuals and better individuals relative to the superior and inferior origin individuals, respectively.

To verify the effectiveness of the IDM strategy, we take the same functions above to compare the performances of three IDMs (denoted by IDM-3T, IDM-5T, and IDM-7T) with different G_T values (i.e., 3T, 5T, and 7T) and the four previous mutation strategies with the same control parameter configuration ($F = 0.5$, CR = 0.5), and add the corresponding convergence curves in Fig. 4.

As the diagrams illustrate, IDMs obtain the best performance in multimodal cases. For unimodal cases, even though the perturbations weaken the convergence rate, the IDMs achieve a competitive performance. Three IDMs have similar performances, and IDM-5T is taken as the mutation strategy of IDE. Combining different origin individuals with corresponding guiders at different stages of the searching process, the new strategy simultaneously improves diversity and accelerates convergence. Moreover, with the perturbations, the IDM strategy is better able to jump out of the local minima to search for the global minimum in multimodal cases.

C. Complete Procedure of the Proposed IDE

Incorporating the novel IDP setting and IDM strategy introduced above, we can now present the whole IDE, following the classic DE framework.

1) Initialization: The initialization of IDE is the same as in classic DE. Generate the initial population and set up the relevant parameters: NP and $g_{\max}$ ($g_{\max}$ = MaxFES/NP).

2) Mutation: For each generation g, all individuals are reindexed in ascending order of their fitness (objective function) values and classified into two sets via (12). Then, the mutant individuals are generated by IDM strategy as (13).

The value of the scale factor is dependent on the origin individual $\mathbf{x}_{o,g}$ in the base vector and is derived from the normal distribution in (5) with a mean value calculated by (1), i.e., rank-based scheme.

3) Crossover: In our approach, the binary crossover operation in classic DE is adopted to generate the trial individuals. The formula of the crossover operation is expressed as Step 2.3 in Algorithm 1.

The value of the crossover rate is dependent on the target individual $\mathbf{x}_{i,g}$ and derived from the normal distribution in (6) with a mean value calculated by (2), i.e., rank-based scheme. Reinitialize the corresponding components of trial individuals that violate the boundary constraints.

4) Selection: As in classic DE, after comparing each target individual with a corresponding trial individual, the individual with the smaller fitness value will enter the next generation.

The pseudo-code of IDE is expressed in Algorithm 1.

Algorithm 1 Pseudo-Code for the IDE

Step 1: Initialization

Set up the maximum generation number $g_{\max}$, generation index $g = 0$ and randomly initialize a population of NP individuals $\mathbf{P}_g = \{\mathbf{x}_{1,g}, \ldots, \mathbf{x}_{NP,g}\}$ with $\mathbf{x}_{\mathrm{i,g}} = \{x^1{}_{i,g}, \ldots, x^D{}_{i,g}\}$ and each individual is uniformly distributed in the range $[\boldsymbol{L}, \boldsymbol{U}]$ with $i = \{1,2,\ldots, NP\}$.

Step 2: Evolution Iteration

WHILE the termination criterion is not satisfied

DO

Step 2.1: Mutation Operation

Reindex the individuals of current population in ascending order of their fitness (objective function) values, classify them into two sets: the superior (S) and the inferior (I) using the proportion model.

FOR $i = 1$ to NP

Randomly choose four different individuals $\mathbf{x}_{o,g}$, $\mathbf{x}_{r1,g}$, $\mathbf{x}_{r2,g}$, and $\mathbf{x}_{r3,g}$ from current population.

Set $o = i$ when $g \le g_t$.

Use rank-based IDP strategy to generate the F_o for the individual $\mathbf{x}_{o,g}$, as:

$F_o = randn(o/NP, 0.1)$.

Generate mutant vector $\mathbf{v}_{i,g}$ via the IDM strategy as:

$$\mathbf{v}_{i,g} = \mathbf{x}_{o,g} + \begin{cases} F_o \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{o,g}) + F_o \cdot (\mathbf{x}_{r_2,g} - \mathbf{d}_{r_3,g}) & o \in S \\ F_o \cdot (\mathbf{x}_{better,g} - \mathbf{x}_{o,g}) + F_o \cdot (\mathbf{x}_{r_2,g} - \mathbf{d}_{r_3,g}) & o \in I \end{cases}$$

$o \neq r_1 \neq r_2 \neq r_3$.

END FOR

Step 2.2: Crossover Operation

FOR $i = 1$ to NP

Employ rank-based IDP strategy to generate $CR_i = randn(i/NP, 0.1)$ for the target individual $\mathbf{x}_{i,g}$. Generate trial vector $\mathbf{u}_{i,g}$ through binomial crossover as follow.

FOR $j = 1$ to D

$$u^j_{i,g} = \begin{cases} v^j_{i,g}, & \text{if}(\text{rand}^j_i(0,1) \le CR_i \text{ or } j = j_{\text{rand}}) \\ x^j_{i,g}, & \text{otherwise} \end{cases}$$

IF ($u^j{}_{i,g} < L^j$ or $u^j{}_{i,g} > U^j$)

$u^j_{i,g} = L^j + rand^j_i(0,1) \cdot (U^j - L^j)$

END IF

END FOR

END FOR

Step 2.3: Selection Operation

Evaluate the trial vector $\mathbf{u}_{i,g}$

FOR $i = 1$ to NP

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{if } (f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g})) \\ \mathbf{x}_{i,g}, & \text{otherwise} \end{cases}$$

END FOR

Step 2.4: Increase the Generation Count

$g = g + 1$

END WHILE

IV. Experimentation

In this section, extensive experiments are carried out to evaluate the IDE performance. We employ the latest 28 test functions (including five unimodal functions and 23 multimodal functions) in the CEC 2013 benchmark suite. Please refer to [48] for complete information and details about the CEC 2013 benchmark suite.

We present the experiment results in four subsections. First, in Section IV-A, we evaluate several IDEs with different strategy configurations through the problems suite to further verify the effectiveness of the proposed strategies. In Section IV-B, we compare the performance of IDE against five state-of-the-art DE variants. Subsequently, we evaluate the performance of IDE against ten up-to-date DE variants in Section IV-C. In Section IV-D, the comparison is performed between IDE and three other well-known EAs. All the evaluation experiments are implemented by MATLAB 7.8 and SPSS 21 on a personal computer with the Intel 2.83 GHz CPU, 4 GB memory, and Windows 7 operating system.

The dimensionality of the problems are 10, 30, and 50 [48], and the corresponding population sizes for IDE are set to 50, 100, and 200, respectively, which are commonly used and suggested in [15], [34], and [49]. To have a reliable and fair comparison, the recommended parameter configurations (including the values of NP, F, CR, and other additional parameters) are set for all competitor algorithms as introduced in their original papers.

For each algorithm to be tested, we use $\mathbf{x}^*$ and $\mathbf{x}'$ to indicate the known optimum solution and the best solution obtained once the search stops after $10^4 \times \mathrm{D}$ function evaluations [48], respectively. Therefore, the fitness error value $F(\mathbf{x}') - F(\mathbf{x}^*)$ is employed to evaluate the algorithms' performance, and the values smaller than 1.0E-08 are taken as zero. Unless otherwise noted, each algorithm is run 51 times [48] in all instances to obtain the numerical results. The mean error (denoted by Mean) of the 51 runs and the corresponding standard deviation (denoted by Std) are calculated and presented in a number of tables. Due to space limitation, these tables are omitted in

the main text of this paper. However, they can be found in the supplementary file of the paper. In the supplementary file, according to the mean error value, we have highlighted the best performance for each benchmark function by shading the background of the cell.

To get a statistical conclusion of the experimental results, the two-sided Wilcoxon rank sum test [50], [51] is applied to assess the significance of the performance difference between IDE and each competitor. Specifically, the null hypothesis in all pairs is that the observations of solutions obtained by both algorithms compared are independent and identically distributed. The value of significance level α is set to be 5% (0.05). When the null hypothesis is rejected, we mark those cases with "+" and "−" to indicate that the competitor algorithm exhibits superior performance relative to IDE and that IDE performs better with significant difference, respectively. We mark cases with "=" if there is no statistically significant difference between the performances of compared algorithms. In addition, to have a more direct evaluation of the performance of these algorithms, the numbers of the three kinds of statistical significance cases (−/=/+) and the number of best results (Number of Best abbreviated as NoB) obtained by the corresponding algorithm for all functions in each test group are listed at the bottom of the numerical results tables. The convergence curves in terms of the mean error values of the best solutions of each algorithm for three multimodal functions used previously with D = 30 are expressed. To plot the curves, the fitness error values are sampled after (0.01, 0.1, 0.2, 0.3, 0.4 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) × MaxFES for each run [48]. The curves of the 30-D sphere function are not considered here because most algorithms can easily find the global optimum solution within a few sampling points.

A. Evaluation on the Strategies in IDE

The new variant DE proposed in this paper includes two core components: IDP and IDM strategies. In this subsection, IDE is compared with its five variants with different configurations to identify the benefits from the IDP and IDM components.

First, to evaluate the IDP setting, we use IDP-r(F,CR), IDP-r(F), and IDP-r(CR) to denote the IDE variants which assign random values uniformly selected from (0,1) for the parameters in parentheses. In other words, the values of control parameters in IDP-r() are no longer associated with the rankings of corresponding individuals.

Second, for the validation of the IDM strategy, we denote the IDE variant without perturbations and the IDE variant with the mutation strategy of classic DE by IDEwo(p) and IDEwoIDM, respectively.

The detailed numerical results for the comparisons of these IDEs are listed in Tables I–III in the supplementary file. To show the performance clearly, we summarize the number of the best results graphically in Fig. 8(a) and list statistical significance results of the comparison between each of the IDE variants and the proposed IDE in Table II. From the histogram in Fig. 8(a), we can observe that IDE achieves the best results in 12, 18, and 15 cases for the 10-D, 30-D, and 50-D problems, respectively. Regarding the total number of the best results obtained by algorithms in three different dimensional groups, the variants IDP-r(F) and IDP-r(CR) yield the best results in 22 (= 11 + 6 + 5) and 25 (= 10 + 9 + 6) cases, respectively, which are more than the number obtained by IDP-r(F,CR), i.e., 17 (= 6 + 6 + 5). Additionally, from Tables I–III in the supplementary file we can see that IDP-r(F) and IDP-r(CR) achieve better mean values than IDP-r(F,CR) in most problems, especially high dimensional cases. It is also clear that, IDEwo(p) performs much better (36 = 12 + 11 + 13) than IDEwoIDM (13 cases). In the comparison results of statistically significant difference listed in Table II, for 10-D problems, it is clear that IDE outperforms the other IDE variants with significant difference in 13, 9, 6, 3, and 23 cases, and it achieves the performance similar to IDP-r(F,CR), IDP-r(F), IDP-r(CR), and IDEwo(p) in most cases. However, relative to these variants, the performance superiority of IDE becomes gradually more significant with the increase of the problems' dimension. For the 30-D problems,

TABLE II
COMPARISON RESULTS OF STATISTICALLY SIGNIFICANT DIFFERENCES OF IDE WITH ALL COMPETITORS ON 10-D, 30-D, AND 50-D PROBLEMS

Algorithms	Dimensions of the Benchmark Functions								
	10-D			30-D			50-D		
	−	=	+	−	=	+	−	=	+
IDP-r(F,CR)	13	15	0	20	8	0	19	6	3
IDP-r(F)	9	15	4	20	7	1	19	8	1
IDP-r(CR)	6	22	0	16	9	3	17	8	3
IDEwo(p)	3	22	3	6	22	0	4	22	2
IDEwoIDM	23	5	0	23	5	0	17	10	1
CoDE	23	5	0	11	9	8	14	6	8
EPSDE	11	13	4	18	5	5	17	6	5
JADE	14	8	6	12	9	7	14	8	6
jDE	19	7	2	16	7	5	19	5	4
SaDE	18	8	2	19	6	3	18	4	6
DE1	15	9	4	14	7	7	14	6	8
DE2	19	4	5	19	4	5	16	6	6
DE3	17	4	7	21	4	3	20	4	4
DE4	18	6	4	23	3	2	21	4	3
DE5	17	8	3	16	8	4	20	2	6
DE6	19	7	2	17	5	6	16	6	6
DE7	15	7	6	18	6	4	15	6	7
DE8	16	4	8	17	5	6	16	5	7
DE9	9	11	8	12	6	10	11	7	10
DE10	20	7	1	23	5	0	24	4	0
CLPSO	16	7	5	23	5	0	24	3	1
CMA-ES	20	3	5	19	4	5	18	4	6
GL-25	22	6	0	24	4	0	23	4	1

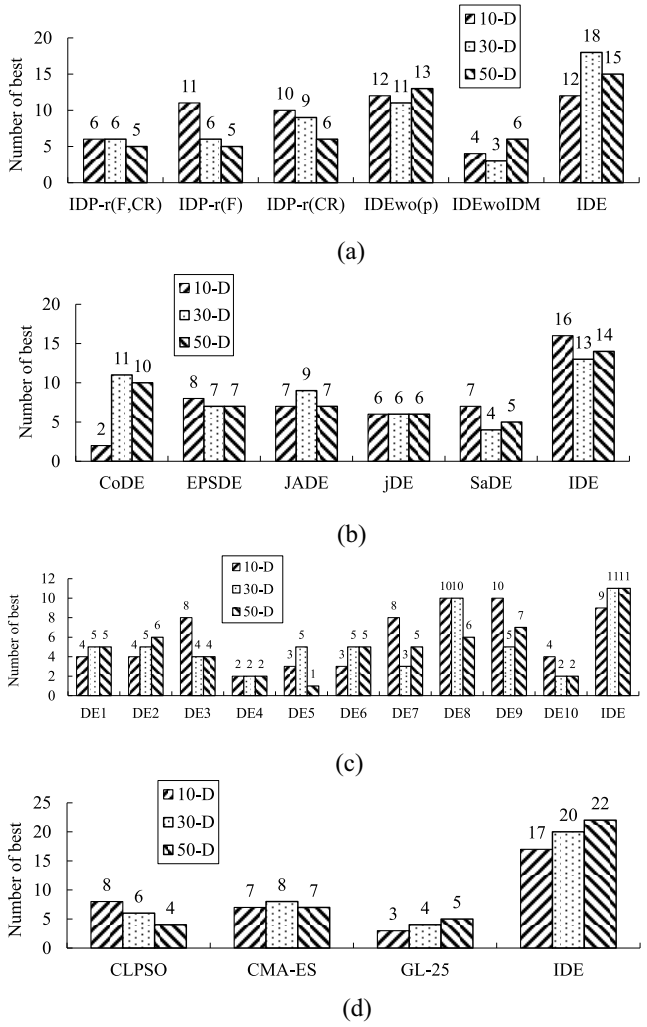

Fig. 8. Number of cases on which each algorithm performs the best in the comparison between (a) different variants of IDE, (b) IDE with five state-of-the-art DEs, (c) IDE with ten up-to-date DEs, and (d) IDE with three EAs.

IDE performs significantly better than IDP-r(*F*,CR), IDP-r(*F*), IDP-r(CR), IDEwo(p), and IDEwoIDM on 20, 20, 16, 6, and 23 functions, respectively. For the 50-D problems, these figures are 19, 19, 17, 4, and 17, respectively. Furthermore, from the convergence curves plotted in Fig. 9(a)–(c), we can observe that IDP-r(*F*) and IDP-r(CR) always converge faster than IDP-r(*F*,CR), and IDEwo(p) outperforms IDEwoIDM on these functions. Despite the fact that IDEwo(p) and IDP-r(CR) achieve better solutions in cases (a) and (c), respectively, there is no statistically significant difference between them and IDE.

B. Comparison With State-of-the-Art DE Variants

In this subsection, we compare the proposed IDE with five state-of-the-art DE variants that have been reported to have good performance.

1) *CoDE:* The DE with composite mutation strategies and parameters [27].
2) *EPSDE:* The DE with an ensemble of parameters and mutation strategies and an SaDE type learning strategy [29].
3) *JADE:* The DE with adaptive control parameters and optional external archive [20].

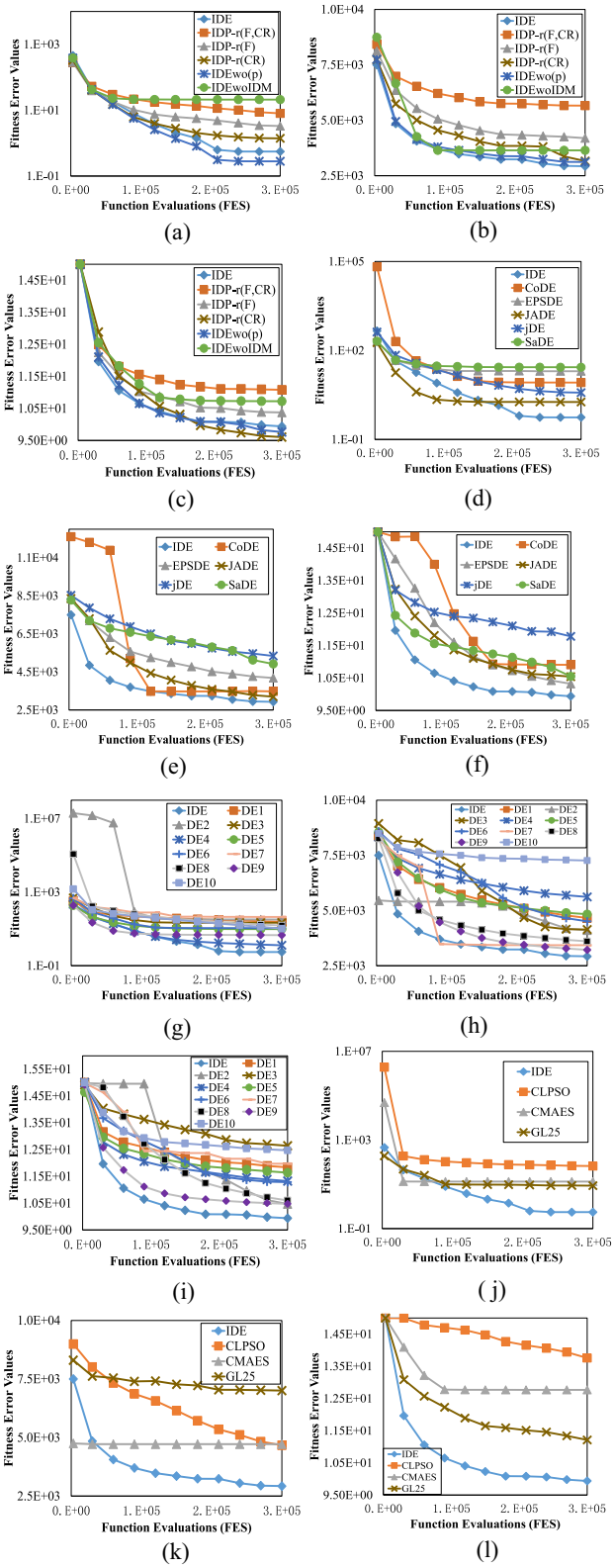

Fig. 9. Convergence curves of (a)–(c) different variants of IDE, different DEs in (d)–(f) Section IV-B and (g)–(i) Section IV-C, and (j)–(l) IDE and different EAs in Section IV-D on three 30-D problems: the Rotated Schaffers F7 function, Schwefel's function, and expanded Scaffer's F6 function, respectively.

4) *jDE:* The DE with self-adaptive control parameters [43].
5) *SaDE:* The DE with adaptive strategy [25].

The experimental results of Number of Best are expressed in Fig. 8(b). Tables IV–VI in the supplementary file report the experimental results for these DE variants in 10-D, 30-D, and 50-D problems, respectively. From the diagram we can observe that for the 10-D problems, IDE obtains the best results in 16 cases, while CoDE, EPSDE, JADE, jDE, and SaDE achieve the best results for 2, 8, 7, 6, and 7 functions, respectively. IDE keeps this obvious advantage in both 30-D and 50-D problems. Moreover, on the 10-D problems the IDE solution is significantly superior to those of CoDE, EPSDE, JADE, jDE, and SaDE on 23, 11, 14, 19, and 18 functions, respectively, as shown in Table II. The comparative results in the reverse direction are only 0, 4, 6, 2, and 2, respectively. For the 30-D and 50-D problems, even though the performance differences between IDE and CoDE are smaller than those for the 10-D problems, the cases of equal performances increase. Additionally, the advantages of IDE over the other DE variants are similar to those for the 10-D problems. From the convergence curves of IDE and state-of-the-art DEs plotted in Fig. 9(d)–(f), we can observe that the best results for the three typical multimodal problems are obtained by IDE.

C. Comparison Against Up-to-Date DE Variants

In this subsection, IDE is compared with nine up-to-date DE variants which are all published on the proceedings of CEC 2013 and one recently proposed Rank-DE.

1) *b6e6rl:* The variant of adaptive DE with twelve competing strategies [46].
2) *SMADE:* The DE with super-fit multicriteria strategy [30].
3) *DE-APC:* The DE with automatic parameter configuration [36].
4) *PVADE:* The DE with population's variance-based adaptive strategy [47].
5) *jDEsoo:* The variant of jDE [31].
6) *SPSRDEMMS:* The structured population size DE with multiple mutation strategies [32].
7) *DEcfbLS:* The DE with concurrent fitness based local search [40].
8) *TLBSaDE:* The DE with teaching and learning best strategy [26].
9) *SHADE:* The adaptive DE with success-history based scheme [45].
10) *Rank-DE:* The improved version of classic DE with ranking-based mutation operator [33].

For the purpose of a simplified presentation,we denote the ten DE variants listed above by DE1–DE10, respectively. Tables VII–IX in the supplementary file report the experimental results for these DE variants in 10-D, 30-D, and 50-D problems, respectively. From the NoB diagram in Fig. 8(c) and the statistical results in Table II, we can observe that for the 10-D problems the proposed IDE yields the best results for 9 of the 28 functions, and performs significantly better than the ten up-to-date DEs on 15, 19, 17, 18, 17, 19, 15, 16, 9, and 20 functions, respectively. When counting the insignificant cases, IDE performs better than or the same as these DEs on 24, 23, 21, 24, 25, 26, 22, 20, 20, and 27 functions, respectively. The comparison results on the 30-D and 50-D problems are similar to those on the 10-D problems, with the proposed algorithm performing even better. From the results, we can see that IDE obtains the best results for 11 cases of the 30-D problems and for 11 cases of the 50-D problems, respectively. While these figures for the latest DEs are fewer than 11 and 8 cases. Of the 28 functions for the 30-D problems, IDE performs better than or the same as the other ten on 21, 23, 25, 26, 24, 22, 24, 22, 18, and 28 functions. For 50-D problems, these figures are 20, 22, 24, 25, 22, 22, 21, 21, 18, and 28. Note that in the convergence curves in Fig. 9(g)–(i), the convergence performance of IDE is the best among these DEs for the three typical functions.

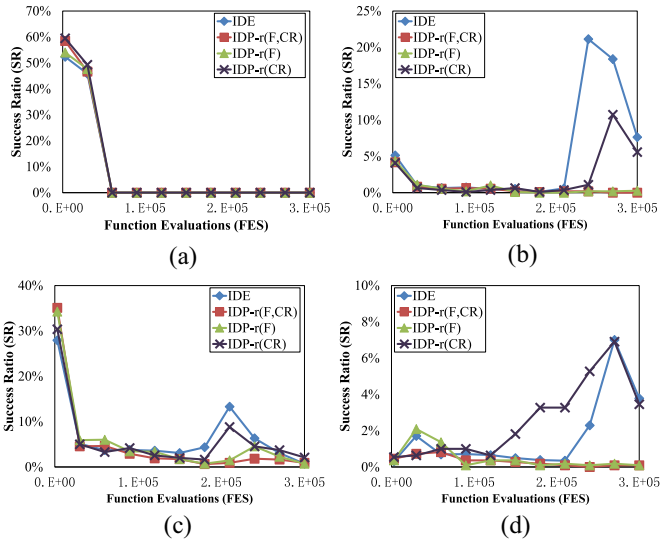

Fig. 10. Success ratio curves of IDE on four 30-D benchmark problems. (a) Sphere function. (b) Rotated Schwefel's function. (c) Rotated Schaffers F7 function. (d) Rotated expanded Scaffer's F6 function.

D. Comparison With NonDE EAs

IDE is further compared with three other EAs.

1) *CLPSO:* The comprehensive learning particle swarm optimizer [52].
2) *CMA-ES:* The evolution strategies with completely derandomized self-adaptation [53].
3) *GL-25:* The global and local real-coded genetic algorithms with parent-centric crossover operators [54].

In CLPSO, introduced by Liang *et al.* [52], the best historical information of each particle is utilized to update the corresponding velocity vector. Hansen and Ostermeier [53] proposed CMA-ES, a powerful variant of evolution strategy (ES). Garcia-Martinez *et al.* [54] presented GL-25, which integrates the global and local search abilities to construct a kind of real-coded genetic algorithm. In our experiments, for all nonDE algorithms, the associated control parameters are set in accordance with their original papers.

The experimental results on the 10-D, 30-D, and 50-D problems are listed in Tables X–XII, respectively, in the supplementary file. Fig. 8(d) provides the results of NoB. For the 10-D problems, IDE achieves the best results in 17 cases, while CLPSO, CMA-ES, and GL-25 achieve the best results in 8, 7, and 3 cases, respectively. IDE significantly outperforms

these three EAs on 16, 20, and 22 of the 28 functions, and performs worse on far fewer functions, as shown in Table II. In addition, IDE shows similar or even better comparative performance for the 30-D and 50-D problems. Similar to the results of the previously described comparisons, IDE performs best in convergence rate as expressed in Fig. 9(j)–(l).

E. Discussion on IDE

For each origin individual in the composite base vector and for each target individual, the IDP setting assigns control parameters based on the differences between fitness values such that the better individuals tend to be assigned relatively smaller values of parameters (both F and CR). Even though the individuals' fitness values and their distance from the optimum is less correlated, especially in multimodal cases, the better individuals are more likely to be found in the neighborhoods close to the superior individuals and in the areas relatively far away from the inferior individuals.

As is evident in the results shown in Section IV-A, in most cases, IDE with the IDP setting performs better than IDP-r(F) and IDP-r(CR), both of which achieve a similar performance and obtain better results relative to the variant IDP-r(F,CR). With the individual-dependent scheme, IDM strategy selects the diversity mutation operators and the rapidity mutation operators for superior individuals and inferior individuals, respectively. Employing the operators with different searching abilities, IDMwo(p) can find better solutions, relative to IDEwoIDM. To further diversify the population, noise elements are introduced in IDM to perturb the difference vectors. Even though IDMwo(p) converges faster than IDE in a few unimodal problems, the latter performs better in multimodal cases. Moreover, the IDE beat all competitor algorithms in the comparison of statistically significant differences.

We plot the success ratio curves of IDEs with different parameter settings on four 30-D pilot functions in Fig. 10. Compared to the SR curves of classic DE expressed in Fig. 6, it is clear that, at the earlier stage, a larger SR value speeds up convergence in the unimodal case [Fig. 10(a)], and at the later stage, an increased SR improves the searching effectiveness in the multimodal cases [Fig. 10(b)–(d)]. More specifically, IDE and IDP-r(CR) achieve increased SR after the transition of mutation operators from the earlier stage to the later stage of the searching process. The generation index threshold vales g_t of IDE on 10-D, 30-D, and 50-D problems are summarized in Table XIII of the supplementary file. We note that the g_t of the functions F1, F5 (except for 50-D), F6 (for 10-D), F21, and F28 (except for 50-D) is smaller than the corresponding G_T. In these functions, the SR values decrease to zero before G_T generations in the cases that finding the global minimum or that falling into the local minima. Because functions F2, F4, F7–F9, F11–F20, F22, and F23 have similar g_t values ($\approx G_T + T$) in each dimension group, it can be assumed that in these cases SR values decrease below SR_T before or at generation G_T. For other functions, IDE maintains a relatively good SR from start to generation g_t. Different problems have different g_t values due to different difficulty degrees. Furthermore, to illustrate the distribution of the results of each algorithm, the box plots of fitness error values of all the algorithms on three 30-D multimodal functions in the pilot experiment are depicted in Fig. 11. From the plots, we can see that IDE performs more consistently than other algorithms on these problems.

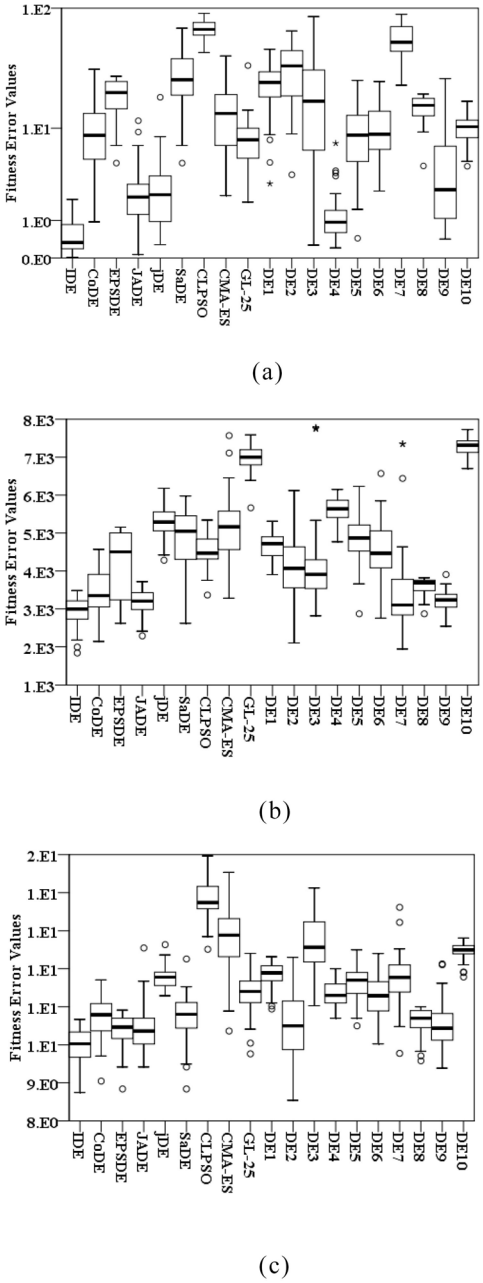

Fig. 11. Comparison results between IDE with other algorithms on three 30-D functions. 30-D rotated (a) Schaffers F7 function, (b) Schwefel's function, and (c) expanded Scaffer's F6 function.

V. Conclusion

According to the literature, many researches have focused on tuning the control parameters and designing the mutation strategies of DE to improve the algorithm performance. However, during the iteration process, most of these methods fail to consider the differences of individuals when assigning the control parameters and selecting the mutation operators.

Generally, the better solutions tend to be found in the neighborhoods of the superior individuals and in the searching

areas relatively far away from the inferior individuals. In this paper, we proposed the individual-dependent mechanism to improve the performance of DE by setting the parameters and choosing the mutation strategies based on differences in individuals' fitness values. At each generation, the individuals in the current population are classified into two categories, i.e., the superior and the inferior, according to their fitness values. Two new IDP setting schemes, namely, the rank-based scheme and the value-based scheme, are employed to calculate the control parameter values for each individual. Then, to generate promising mutant individuals, the proposed IDE employs a new IDM strategy. In IDM, for the purpose of balancing diversity with rapidity, the random individuals and the better individuals are employed to guide the searching directions of the superior individuals and the inferior individuals, respectively. In addition, the current and the random individuals are taken as the origin individuals in the composite base vectors at two different stages of the iteration. To the best of our knowledge, this represents the first attempt to modify DE using the information derived from the differences between the fitness values of individuals.

To evaluate the performance of the proposed algorithm, extensive experiments were carried out on the latest 28 test functions in the CEC 2013 benchmark suite. Considering the No Free Lunch theorem [55], there is no algorithm that can obtain the best performance for all benchmark problems. In comparison with 18 excellent algorithms, IDE achieves both significantly superior and inferior performances, but clearly the superior outweighs the inferior.

As a continuation of this research, we are investigating the discrete version of DE based on the individual-dependent scheme. We are also identifying opportunities to solve practical optimization problems.

The source codes in MATLAB for the competitor algorithms in Sections IV-B (five state-of-the-art DEs) and IV-D (three other EAs) can be downloaded from the website http://dces.essex.ac.uk/staff/qzhang/ [27], and the MATLAB codes of the benchmark functions suite and the detailed raw result data of the first nine up-to-date DEs in Section IV-C are available on http://www.ntu.edu.sg/home/epnsugan/ [48]. The supplementary file of this paper can be downloaded from Tang's homepage: http://ielo.neu.edu.cn.

REFERENCES

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

[2] R. Storn and K. V. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 842–844.

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

[4] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*. Berlin, Germany: Springer, 2005.

[5] J. Zhang, V. Avasarala, A. C. Sanderson, and T. Mullen, "Differential evolution for discrete optimization: An experimental study on combinatorial auction problems," in *Proc. IEEE World Congr. Comput. Intell.*, Hong Kong, Jun. 2008, pp. 2794–2800.

[6] J. Zhang, V. Avasarala, and R. Subbu, "Evolutionary optimization of transition probability matrices for credit decision-making," *Eur. J. Oper. Res.*, vol. 200, no. 2, pp. 557–567, Jan. 2010.

[7] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact differential evolution," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 32–54, Feb. 2011.

[8] B. V. Babu and R. Angira, "Modified differential evolution (MDE) for optimization of non-linear chemical processes," *Comput. Chem. Eng.*, vol. 30, nos. 6–7, pp. 989–1002, 2006.

[9] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, May 2010.

[10] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 218–236, Jan. 2008.

[11] N. Noman and H. Iba, "Inferring gene regulatory networks using differential evolution with local search heuristics," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 4, no. 4, pp. 634–647, Oct. 2007.

[12] P. P. Menon, J. Kim, D. G. Bates, and I. Postlethwaite, "Clearance of nonlinear flight control laws using hybrid evolution optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 689–699, Dec. 2006.

[13] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 209–225, Apr. 2014.

[14] 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.

[15] 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.

[16] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, no. 1, pp. 61–106, Feb. 2010.

[17] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[18] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Balancing the exploration and exploitation capabilities of the differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Hong Kong, Jun. 2008, pp. 2686–2693.

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

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

[21] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[22] C. Lin, A. Qing, and Q. Feng, "A new differential mutation base generator for differential evolution," *J. Global Optim.*, vol. 49, no. 1, pp. 69–90, Feb. 2011.

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

[24] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Sep. 2005, pp. 1785–1791.

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

[26] S. Biswas, S. Kundu, S. Das, and A. V. Vasilakos, "Teaching and learning best differential evolution with self adaptation for real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1115–1122.

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

[28] 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, Mar. 2011.

[29] S. Z. Zhao and P. N. Suganthan, "Comprehensive comparison of convergence performance of optimization algorithms based on nonparametric statistical tests," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, Jun. 2012, pp. 1–7.
[30] F. Caraffini *et al.*, "Super-fit multicriteria adaptive differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1678–1685.
[31] J. Brest, B. Boškoviæ, A. Zamuda, I. Fister, and E. Mezura-Montes, "Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 337–383.
[32] A. Zamuda, J. Brest, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1925–1931.
[33] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2066–2081, Dec. 2013.
[34] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Edinburgh, U.K., 2005, pp. 506–513.
[35] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Inf. Sci.*, vol. 185, no. 1, pp. 153–177, 2012.
[36] S. M. Elsayed, R. A. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1932–1937.
[37] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. Genet. Evol. Comput. (GECCO)*, Jun. 2005, pp. 991–998.
[38] H. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Honolulu, HI, USA, May 2002, pp. 831–836.
[39] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proc. Comput. Intell. Secur.*, 2005, pp. 192–199.
[40] I. Poikolainen and F. Neri, "Differential evolution with concurrent fitness based local search," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 384–391.
[41] F. Neri, G. Iacca, and E. Mininno, "Disturbed exploitation compact differential evolution for limited memory optimization problems," *Inf. Sci.*, vol. 181, no. 12, pp. 2469–2487, 2011.
[42] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput*., vol. 9, no. 6, pp. 448–462, 2005.
[43] 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, Dec. 2006.
[44] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.
[45] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 71–78.
[46] J. Tvrdík and R. Poláková, "Competitive differential evolution applied to CEC 2013 problems," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1651–1657.
[47] L. S. Coelho, H. V. H. Ayala, and R. Z. Freire, "Population's variance-based adaptive differential evolution for real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1672–1677.
[48] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, Jan. 2013. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/
[49] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[50] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull*., vol. 1, no. 6, pp. 80–83, Dec. 1945.
[51] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 1947.
[52] 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. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[53] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
[54] 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," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1088–1113, Mar. 2008.
[55] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

Lixin Tang (M'09–SM'14) received the B.Eng. degree in industrial automation, the M.Eng. degree in systems engineering, and the Ph.D. degree in control theory and application from Northeastern University, Shenyang, China, in 1988, 1991, and 1996, respectively.

He is a Cheung Kong Scholars Chair Professor and Director of the Institute of Industrial Engineering and Logistics Optimization, Northeastern University. His research interests include batching and scheduling of the production operations, logistics planning and scheduling, inventory control and supply chain planning, production-process operations optimization, and optimal control. His research papers have appeared in academic journals such as *Operations Research*, *IIE Transactions*, *Naval Research Logistics*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON POWER SYSTEMS, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and *European Journal of Operational Research*.

Dr. Tang received the National Natural Science Foundation for Distinguished Young Scholars of China and the State Youth Science and Technology Award of China. He is an Area Editor of *Asia-Pacific Journal of Operational Research*.

Yun Dong is currently working toward the Ph.D. degree in system engineering from Institute of Industrial Engineering and Logistics Optimization, Northeastern University, Shenyang, China.

His research interests include logistics planning and scheduling, modeling and optimization in container terminal operations, decision support systems, and metaheuristics.

Jiyin Liu received the B.Eng. degree in industrial automation and the M.Eng. degree in systems engineering from Northeastern University, Shenyang, China, in 1982 and 1985, respectively, and the Ph.D. degree in manufacturing engineering and operations management from University of Nottingham, Nottingham, U.K., in 1993.

He is a Professor of Operations Management with the School of Business and Economics, Loughborough University, Leicestershire, U.K. He is also a Cheung Kong Scholars Visiting Chair Professor with the Logistics Institute, Northeastern University. His research interests include operations planning and scheduling problems in production; logistics and supply chains; and mathematical modeling, optimization, and heuristic methods. He has published papers in journals such as *European Journal of Operational Research*, *IIE Transactions*, *Naval Research Logistics*, *Operations Research*, and *Transportations Research*.