Abstract Convex Underestimation Assisted Multistage Differential Evolution

Xiao-Gen Zhou and Gui-Jun Zhang

***Abstract*—In differential evolution (DE), different strategies applied in different evolutionary stages may be more effective than a single strategy used in the entire evolutionary process. However, it is not trivial to appropriately determine the evolutionary stage. In this paper, we present an abstract convex underestimation-assisted multistage DE. In the proposed algorithm, the underestimation is calculated through the supporting vectors of some neighboring individuals. Based on the variation of the average underestimation error (*UE*), the evolutionary process is divided into three stages. Each stage includes a pool of suitable candidate strategies. At the beginning of each generation, the evolutionary stage is first estimated according to the average *UE* of the previous generation. Subsequently, a strategy is automatically chosen from the corresponding candidate pool to create a mutant vector. In addition, a centroid-based strategy which utilizes the information of multiple superior individuals is designed to balance the population diversity and convergence speed in the second stage. Experiments are conducted on 23 widely used test functions, CEC 2013, and CEC 2014 benchmark sets to demonstrate the performance of the proposed algorithm. The results reveal that the proposed algorithm exhibits better performance compared with several advanced DE variants and some non-DE approaches.**

***Index Terms*—Abstract convex, differential evolution (DE), evolutionary algorithm (EA), global optimization, underestimation.**

I. Introduction

DIFFERENTIAL evolution (DE), as one of the most efficient stochastic optimization approaches for global optimization, is first introduced by Storn and Price [1]. Inspired by the biology evolutionary process, DE guides the searching process based on the intelligence produced by the competition and cooperation among individuals in the population. Owing to its ease of use, efficiency, robustness, and speed, DE has been effectively employed to solve a wide variety of practical problems, such as economic dispatch [2], aircraft control [3], protein folding [4], and sensor network localization [5]. A detailed survey on other applications and researches of DE can be found in [6] and [7].

Similar to other evolutionary algorithms (EAs), such as genetic algorithm [8], evolution strategies (ES) [9], and particle swarm optimization (PSO) [10], DE also includes three evolutionary operators, namely, mutation, crossover, and selection. The differential mutation is a core operator among these three operators [11]. The task of mutation is to generate a new individual via combining the randomly selected or prespecified individuals. The main objective of mutation is to increase the population diversity and consequently preventing the premature convergence. Over the past decades, many mutation strategies, such as DE/rand-to-best/1, DE/lbest/1 [12], DE/current-to-*p*best/1 [13], and ranking-based mutation [11], have been developed for DE. Generally, different strategies display distinct characteristics and are suitable for different stages of the evolutionary process. Therefore, different stages employ different strategies may be more effective than the entire evolutionary process uses a single strategy.

Recently, several approaches are presented to enhance the search capability of DE by dividing the evolutionary process into multiple stages and selecting different mutation strategies for each stage of the evolutionary process. Fan and Yan [14] introduced a DE with zoning evolution of control parameters and adaptive mutation strategies (ZEPDE). In ZEPDE, the evolutionary process is divided into two stages according to a given maximum generation. The strategy DE/rand/1 is employed to generate offspring individuals in the first stage. In the second stage, mutation strategies are automatically selected by using a roulette wheel in light of the cumulative selective probability of each strategy. Cheng and Tran [15] proposed a two-phase DE. On the basis of a given maximum generation, the search process is divided into three equally sized phases, and some suitable mutation strategies are employed in each phase. Experimental results have demonstrated that the above approaches can enhance the search capability of DE. However, these two evolutionary stage division methods highly rely on the maximum generation, which cannot be obtained in advance for a specific problem. Different problems usually require different generations to find the global optimum, and some problems may not reach certain stages within a given maximum generation. Inappropriate mutation strategies used in these stages may lead to premature convergence, thus influencing the quality of the final solution.

In order to divide the evolutionary process into several more reasonable stages and further enhance the performance

Manuscript received March 11, 2017; accepted May 27, 2017. Date of publication June 8, 2017; date of current version August 16, 2017. This work was supported in part by the National Nature Science Foundation of China under Grant 61075062 and Grant 61573317, and in part by the China Scholarship Council. This paper was recommended by Associate Editor P. N. Suganthan. *(Corresponding author: Gui-Jun Zhang.)*

The authors are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: zgj@zjut.edu.cn).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

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/TCYB.2017.2710626

of DE, an abstract convex underestimation-assisted multistage DE (UMDE) is proposed in this paper. In UMDE, the underestimation value of each trail individual is first calculated by the abstract convex supporting vectors of some neighboring individuals based on abstract convexity theory [16], [17]. According to the variation of the average underestimation error (*UE*) of all trial individuals, the searching process is partitioned into three different evolutionary stages. Each stage has its own strategy candidate pool, and the strategy is adaptively determined during the evolutionary process. Moreover, a centroid-based strategy is designed to get a tradeoff between the population diversity and convergence speed in the second stage. The performance of UMDE is compared with several advanced DE methods and some non-DE approaches over 23 frequently used benchmark functions, CEC 2013, and CEC 2014 benchmark sets.

The remainder of this paper is organized as follows. Section II briefly describes the basic DE and abstract convex underestimation. Section III presents a short literature review on DE. In Section IV, the proposed UMDE is described in detail. Experimental results and parameters analysis of UMDE are reported in Section V. Section VI concludes this paper.

II. Preliminary

A. Differential Evolution

Suppose that the problem to be minimized is $f(\boldsymbol{x}), \boldsymbol{x} \in R^D$ in this paper. The procedure of DE is described as follows.

1) *Initialization:* Randomly produce an initial population $P_0 = \{\boldsymbol{x}_i^0 = (x_{i,1}^0, x_{i,2}^0, \ldots, x_{i,D}^0), i = 1, 2, \ldots, NP\}$ from the solution domain, where NP is the population size. Determine the value of scaling factor F, crossover rate CR, and maximum generation $G_{\max}$. The generation count g is set as 0.
2) For each target individual $\boldsymbol{x}_i^g, i = 1, 2, \ldots, NP$ in the current population, execute steps 3)–5) to generate individuals for the next generation.
3) *Mutation:* A mutation vector $\boldsymbol{v}_i^g$ is created in the light of the following way:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{r_1}^g + F \cdot \left(\boldsymbol{x}_{r_2}^g - \boldsymbol{x}_{r_3}^g\right) \tag{1}$$

where $F \in (0, 1]$ is the scaling factor, r_1, r_2, and $r_3 \in [1, NP]$, and $r_1 \neq r_2 \neq r_3 \neq i$.
4) *Crossover:* A trial vector $\boldsymbol{u}_i^g = (u_{i,1}^g, u_{i,2}^g, \ldots, u_{i,D}^g)$ is created as follow:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } R_j \leq CR \text{ or } j = j_{\text{rand}} \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{2}$$

where $j = 1, 2, \ldots, D$, j_{rand} is an randomly generated integer within the range $[1, D]$; R_j is a uniform distributed random number chosen from the range $[0, 1]$ for each j; and $CR \in (0, 1)$ is the crossover rate.
5) *Selection:* According to the fitness value, determine the better one between the trial vector u_i^g and the target vector x_i^g in the following way:

$$\boldsymbol{x}_i^{g+1} = \begin{cases} \boldsymbol{u}_i^g, & \text{if } f\left(\boldsymbol{u}_i^g\right) \leq f\left(\boldsymbol{x}_i^g\right) \\ \boldsymbol{x}_i^g, & \text{otherwise.} \end{cases} \tag{3}$$

6) $g = g + 1$ and repeat steps 2)–5) until g is larger than the maximum generation $G_{\max}$.

Remark: Some other widely used mutation strategies are given in the supplementary file due to the page limitation.

B. Abstract Convex Underestimation

Abstract convexity theory [16], [17] is generalized from some properties of convex analysis that each convex function can be represented as the upper envelop of its affine minorants [18]. Based on this conclusion, an underestimation of the objective function always can be obtained by using some simple functions. Suppose there are K given points $(\boldsymbol{x}^k, f(\boldsymbol{x}^k)), k = 1, 2, \ldots, K$, the supporting vector of each point can be calculated according to abstract convexity theory [16], [17], namely

$$\boldsymbol{l}^k = \left(\frac{f\left(\boldsymbol{x}^k\right)}{M} - x_1^k, \frac{f\left(\boldsymbol{x}^k\right)}{M} - x_2^k, \ldots, \frac{f\left(\boldsymbol{x}^k\right)}{M} - x_{D+1}^k\right) \tag{4}$$

where $\boldsymbol{x}^k$ is the kth supporting point, $x_{D+1}^k = 1 - \sum_{j=1}^{D} x_j^k$ is a slack variable, and M is a constant to control the slope of the supporting vector [19]. Based on these K supporting vectors, a piecewise linear underestimation of the objective function can be obtained by

$$U(\boldsymbol{x}) = M \max_{k \leq K} \min_{j=1,\ldots,D+1} \left(l_j^k + x_j\right). \tag{5}$$

The underestimation value of each point in the feasible region can be calculated according to (5).

Fig. S-2 (see supplementary material) illustrates the underestimation of a 1-D objective function. As shown in the figure, the underestimation value of each point in the feasible region can be calculated according to the underestimation $U(\boldsymbol{x})$. In addition, the accuracy of the underestimation is considerably related to the density of the supporting points. As shown in the green region, the underestimation is more accurate than the other region as more supporting points are included in this region. Detailed description of the abstract convex underestimation can be found in the supplementary material.

III. Literature Review

Since many mutation strategies are developed for DE, it is not easy to select a best one from these strategies for a specific problem as different strategy has different characteristic. In order to overcome this weakness, many approaches, such as strategies self-adaptive techniques and new mutation strategies are developed in the last few decades. We briefly review these enhanced approaches in this section.

Some work mainly focuses on adaptively choosing different mutation strategies for different stages from a strategy candidate pool. Qin *et al.* [20] presented a strategy self-adaptive DE (SaDE). In SaDE, the previous successful experience is applied to adaptively choose strategies for different stages from the strategy candidate pool. Wang *et al.* [21] introduced a composite DE (CoDE), in which several trial individuals are simultaneously produced via randomly combining the mutation strategy with the control parameter in the respective candidate pool, and the best one is selected as the offspring

individual. Gong *et al.* [22] described a strategy adaptation mechanism (SaM). In this method, mutation strategies are chosen from the strategy candidate pool by using a adaptively updated strategy parameter η. Mallipeddi *et al.* [23] proposed a DE using ensemble of mutation strategies and parameters (EPSDE), in which the control parameters and mutation strategy are randomly chosen from the corresponding candidate pool to produce the offspring individual. Gong *et al.* [24] proposed another two different mechanisms (i.e., adaptive pursuit and probability matching) to select the strategy adaptively according to their recent impact calculated based on the improved fitness. Pan *et al.* [25] presented a DE with self-adaptive trial vector generation strategy and control parameters (SspDE). In SspDE, strategies that generate better individuals are saved in a winning strategy list. The strategy of each target individual is chosen from the associated strategy list which is refilled by selecting elements from the winning strategy list after a given number of iterations. Gong *et al.* [26] presented a multioperator search strategy based on the cheap surrogated model. In this technique, a set of offspring individuals is created by several different mutation strategies, and one offspring individual is selected according to their estimated probabilities obtained from the surrogate model. Elsayed *et al.* [27] described a DE with multiple strategies, which divides the population into four distinct groups. For each individual in the same group, one type of mutation strategy is applied to generate a trial individual. Wu *et al.* [28] proposed a multipopulation ensemble DE (MPEDE). MPEDE dynamically divides the population into one reward subpopulation and three indicator subpopulations. The reward subpopulation and the indicator subpopulation are allocated to the best performing mutation strategy and the constituent mutation strategy, respectively.

Many new mutation strategies have also been developed to balance the exploration and exploitation capabilities of DE. Zhang and Sanderson [13] presented an adaptive DE with optional external archive (JADE). In JADE, a new mutation strategy is introduced, referred as DE/current-to-*p*best/1. In the new strategy, a solution is randomly selected from the top $p\%$ solutions rather than the global best solution found so far. Yu *et al.* [12] also introduced an adaptive DE (ADE), in which another new mutation strategy named DE/1best/1 is designed. In ADE, the population is divided into some equally sized groups. The locally best solution in each group is chosen to replace the best solution in DE/best/1. Gong and Cai [11] described a ranking-based mutation strategy, in which the base vector and one of the terminal vectors are selected according to a probability calculated by their rankings of the fitness values. Cai and Wang [29] designed a direction induced mutation strategy, in which the base and different vectors are determined by a neighborhood guided selection scheme. Zhou *et al.* [30] described an intersect mutation strategy. At each generation, the population is divided into the better and worse parts according to the fitness value. The base vector is randomly chosen from the better part, while the differential vectors are randomly selected from the worse part. Guo *et al.* [31] also proposed a new mutation strategy, in which parents are chosen from the successful solutions stored in an archive if a solution is not improved for some continuous iterations. Wang *et al.* [32] introduced a Gaussian mutation strategy without scaling factor F, where the mutant vector is generated according to a Gaussian random function. Das *et al.* [33] proposed an improved variant of "current-to-best/1" strategy based on a local neighborhood model, in which the best vector and the other two vectors are chosen from the small neighborhood of each target vector. Epitropakis *et al.* [34] presented a proximity-based mutation strategy. In this strategy, the parent individual is determined according to a probability calculated according to the distance between each individual and the mutated individual.

IV. UMDE Algorithm

This section introduces the proposed UMDE algorithm. UMDE is characterized by an abstract convex underestimation-based stage estimation scheme and a new centroid-based mutation strategy. In the abstract convex underestimation-based stage estimation, the average *UE* between the actual function value and the underestimation value obtained from the abstract convex supporting vectors is used to estimate the evolutionary stage. For each stage, some suitable strategies are employed to generate offspring individuals. In addition, the centroid-based mutation strategy is designed to balance the population diversity and fast convergence in the second stage.

A. Abstract Convex Underestimation-Based Stage Estimation

As mentioned in Section II, an underestimation of the objective problem always can be obtained based on the supporting vectors of the given points [17], [35]. The accuracy of the underestimation is considerably related to the density of supporting points that are used to construct the supporting vectors. The denser supporting points often correspond to the smaller *UE*.

To determine the evolutionary stage of DE by using the abstract convex underestimation information, each individual of the population is regarded as a supporting point, and its supporting vector is calculated to obtain the underestimation of the objective function. In the initial population, the value of *UE* is relatively large since the individual is comparatively dispersive. As the evolution proceeds, the value of *UE* gradually decreases with the convergence of the population. At the later stage, as the population finally clusters around a global or local optimum, the value of *UE* is relatively small. Based on the above analysis, we can estimate the evolutionary stage of DE according to the variation of *UE*. Fig. 1 illustrates the variation of the underestimation along with the population convergence for 1-D Ackley's function after 0, 5, 10, and 20 generations.

From the above analysis, we can know that the underestimation information can be employed to estimate the evolutionary stage of DE. However, calculating the supporting vectors of all individuals at each generation may lead to high computational complexity. In order to reduce the computational complexity, in the proposed UMDE, we only calculate the supporting vectors of the n individuals near the trial individual (measured by Euclidean distance). Then, the average *UE* between the underestimation value obtained from the supporting vectors

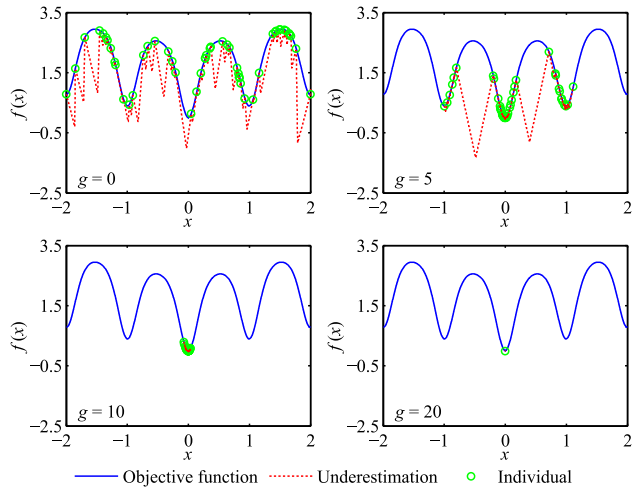


Fig. 1. Underestimation after 0, 5, 10, and 20 generations.

and the actual function value of all trial individuals is used for the evolutionary stage estimation. In addition, all supporting vectors are removed after each iteration to release the computer memory. The detailed steps of the evolutionary stage estimation are as follows.

Step 1: For each trial individual ($\boldsymbol{u}_i^g$, $i = 1, 2, \ldots, NP$) in each generation, compute Euclidean distance from it to each individual of the population. Sort the population in an ascending order according to their distances. The top n individuals are selected and denoted as $\boldsymbol{x}_t^{\text{near},g}$, $t = 1, 2, \ldots, n$.

Step 2: Calculate the supporting vectors of the n selected individuals

$$\boldsymbol{l}_t^{\text{near},g} = \left(\frac{f\left(\boldsymbol{x}_t^{\text{near},g}\right)}{M} - x_{t,1}^{\text{near},g}, \ldots, \frac{f\left(\boldsymbol{x}_t^{\text{near},g}\right)}{M} - x_{t,D+1}^{\text{near},g} \right) \quad (6)$$

where $x_{t,D+1}^{\text{near},g} = 1 - \sum_{j=1}^{D} x_{t,j}^{\text{near},g}$ is the slack variable, and M is the slope control factor of the supporting vector [19].

Step 3: Compute the underestimation value ($U(\boldsymbol{u}_i^g)$) of the trial individual $\boldsymbol{u}_i^g$ according to the supporting vectors

$$U\left(\boldsymbol{u}_i^g\right) = M \max_{t \le n} \min_{j=1,\ldots,D+1} \left(l_{t,j}^{\text{near},g} + u_{i,j}^g \right) \quad (7)$$

where $l_{t,j}^{\text{near},g}$ is the jth variable of $\boldsymbol{l}_t^{\text{near},g}$.

Step 4: After calculating the underestimation values of the NP trial individuals in the current generation by repeating steps 1–3, compute the average UE of the current generation

$$UE = \frac{1}{NP} \sum_{i=1}^{NP} \left| U\left(\boldsymbol{u}_i^g\right) - f\left(\boldsymbol{u}_i^g\right) \right|. \quad (8)$$

In the initial generation, the average UE is comparatively large because the population is relatively dispersive. Therefore, UE of the initial generation is considered as the maximum UE ($UE_{\max}$), and $UE_{\max}$ will be updated by a larger UE during the evolution process. If the population converges to the global or a local optimum, the underestimate error will be equal to 0. At this state, UE has its minimum value $UE_{\min} = 0$.

Step 5: According to the difference between the values of $UE_{\max}$ and $UE_{\min}$, the value of UE is normalized as

$$\overline{UE} = \frac{UE - UE_{\min}}{UE_{\max} - UE_{\min}} = \frac{UE}{UE_{\max}} \quad (9)$$

where $\overline{UE} \in [0, 1]$ is the normalized value of UE.

Step 6: According to the value of $\overline{UE}$, the evolutionary process can be divided into three stages

$$\Psi = \begin{cases} S_1, & \text{if } \overline{UE} \ge \mu \\ S_2, & \text{if } 1 - \mu \le \overline{UE} < \mu \\ S_3, & \text{otherwise} \end{cases} \quad (10)$$

where Ψ represents the estimated evolutionary stage; S_1, S_2, and S_3, respectively denote the first, second, and third stage; and $\mu \in (0.5, 1.0)$ is the stage control parameter.

The division of the evolutionary stage is based on the following considerations. At the first stage S_1, all individuals are scattered in the searching space to explore the promising regions. This stage should be relatively short as DE has a strong exploration capability [36], [37], and it can quickly locate promising regions. In the second stage S_2, the algorithm begins to exploit the explored regions and continues to explore potential promising regions. This stage should be longer to ensure that the explored regions are fully exploited, and more potential regions are explored. At the last stage S_3, all individuals may be attracted into one promising region, and they will be devoted to search the best solution in this region. This stage may take relatively little time in searching only in one region.

B. Centroid-Based Mutation Strategy

As is well known, the best-so-far solution-based strategies, such as "DE/best/1/bin," "DE/rand-to-best/1/bin," and "DE/current-to-best/1/ bin" fast at exploitation of the solution [20]. In these strategies, the best solution found so far is used to guide the search. Due to the information of the best solution, these strategies are good at exploiting the solution. However, the information of the best solution may also reduce the population diversity and the exploration ability of exploring new promising regions. Therefore, individuals are more likely to be attracted into a local minimum.

In order to get a tradeoff between the population diversity and fast convergence, a new centroid-based mutation strategy is presented. At the beginning of each generation, the population is first sorted in descending order based on the fitness value. Then N different individuals are randomly selected from the top $NP/2$ individuals at each iteration to calculate the centroid individual

$$\boldsymbol{x}_{\text{centroid},j}^g = \frac{\sum_{i=1}^{N} x_{i,j}^g f\left(\boldsymbol{x}_i^g\right)}{\sum_{i=1}^{N} f\left(\boldsymbol{x}_i^g\right)} \quad (11)$$

where $j = 1, 2, \ldots, D$; $N \le NP/2$ is called the centroid control parameter; and $x_{\text{centroid},j}^g$ represents the jth variable of the centroid individual $\boldsymbol{x}_{\text{centroid}}^g$.

Three new strategies are presented by replacing the best vector with the centroid vector in DE/best/2, DE/current-to-best/1, and DE/rand-to-best/1.

1) *DE/centroid/2*

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{\text{centroid}}^g + F \cdot \left(\boldsymbol{x}_{r_1}^g - \boldsymbol{x}_{r_2}^g\right) + F \cdot \left(\boldsymbol{x}_{r_3}^g - \boldsymbol{x}_{r_4}^g\right). \quad (12)$$

2) *DE/current-to-centroid/1*

$$\boldsymbol{v}_i^g = \boldsymbol{x}_i^g + F \cdot \left(\boldsymbol{x}_{\text{centroid}}^g - \boldsymbol{x}_i^g\right) + F \cdot \left(\boldsymbol{x}_{r_1}^g - \boldsymbol{x}_{r_2}^g\right). \quad (13)$$

3) *DE/rand-to-centroid/1*

$$\boldsymbol{v}_i^g = \boldsymbol{x}_{r_1}^g + F \cdot \left(\boldsymbol{x}_{\text{centroid}}^g - \boldsymbol{x}_{r_1}^g\right) + F \cdot \left(\boldsymbol{x}_{r_2}^g - \boldsymbol{x}_{r_3}^g\right). \quad (14)$$

Since the centroid vector is inspired by the centroid formula in Physics; hence, we call the above strategies centroid-based strategies. These three new strategies will be used in the second stage of UMDE to balance the population diversity and fast convergence. As shown in (11), the centroid individual is calculated based on the fitness value of each selected individual. For the centroid individual, the characteristic from each selected individual is determined by a weighted value calculated according to the fitness value. Hence, only the partial information of each selected individual is inherited by the centroid individual. As the information from the multiple selected superior individuals rather than the single globally best individual are applied to guide the evolutionary process, the population diversity reduction can be alleviated.

C. Strategy Candidate Pool of Each Stage

Based on the characteristic of each mutation strategy studied in [1], [20], [26], and [38], the strategy candidate pool of each evolutionary stage is established as follow. In the first stage, all individuals are dedicated to exploring different promising regions. Therefore, in this stage, DE/rand/1/bin, DE/current-to-rand/1, and DE/rand/2/bin are selected to preserve the population diversity and ensure that a sufficient number of promising regions are explored. In the second stage, the proposed DE/centroid/2/bin, DE/rand-to-centroid/1/bin, and DE/current-to-centroid/1/bin are employed to balance the exploration and exploitation. Hence, these explored regions in the first stage can be sufficiently exploited, and some other more potential regions are explored. At the last stage, the population may fall into one promising region. The algorithm will pay much attention to improving the best-so-far individuals. Therefore, the greedy strategy, i.e., DE/best/2/bin, DE/rand-to-best/1/bin, and DE/current-to-best/1/bin can be used to enhance the convergence speed in this stage.

A selection probability is allocated for each mutation strategy in each strategy candidate pool. The probability of each mutation strategy in the same pool is initialized as $1/3$ at the first generation of each evolutionary stage. At the end of each generation, the selection probability of the kth strategy is recalculated by its number of successful runs (NS_k) divided by the total number of successful runs of all strategies in the same strategy pool. Here, the successful run means that the trial vector gets a better fitness value compared to its corresponding target vector. To avoid the possible null selection probabilities, the selection probability of each strategy is added to a small constant value $\varepsilon = 0.01$ as suggested in [20]. As the evolutionary stage is determined by the *UE* of the previous generation, DE/rand/1/bin is employed in the initial generation. At the beginning of each generation after the initial generation, the evolutionary stage is first determined by (10). Then a mutation strategy in the corresponding strategy pool is selected by using a roulette wheel based on the selection probability of each strategy.

D. Parameter Adaptation

In the aforementioned DE variants in Section III, many approaches have been presented to choose the values of F and CR adaptively or self-adaptively. Inspired by these approaches, a simple strategy is designed to update F and CR dynamically in this paper. Like some famous approaches, such as SaDE [20] and JADE [13], the crossover rate CR_i^g in the proposed UMDE is also independently calculated for each individual by a normal distribution with mean value CRm^g and standard deviation 0.1, denoted as $N(CRm^g, 0.1)$, where CRm^g is initialized to be 0.5. To update the value of CRm^g, memories named successful CR (SCR) are established to store the CR_i^g values which have generated better offspring solutions and the corresponding improvement fitness values IF_i^g in the current generation. Assuming that there are NS^g combinations stored in SCR at the gth generation, then a weighted mean (W_{CR}^g) can be calculated by

$$W_{CR}^g = \frac{\sum_{s=1}^{NS^g} CR_s^g \cdot IF_s^g}{\sum_{s=1}^{NS^g} IF_s^g} \quad (15)$$

where $IF_s^g = |f(\boldsymbol{u}_s^g) - f(\boldsymbol{x}_s^g)|$. If there is no CR successfully generates better offspring solutions at the current generation, then $W_{CR}^{g+1} = W_{CR}^g$. According to each W_{CR}^g in the previous learning generations (*LGs*), the value of CRm^g is updated by

$$CRm^{g+1} = \frac{\sum_{G=g-LG}^{g-1} W_{CR}^G \cdot NS^G}{\sum_{G=g-LG}^{g-1} NS^G}. \quad (16)$$

In the first LG generations, CRm^g is replaced by W_{CR}^g. As Qin *et al.* suggested in [20], the value of LG is set to 20 in this paper.

Similarly, for each individual, the scaling factor F_i^g is independently created by a Cauchy distribution $C(Fm^g, 0.1)$, where Fm^g (initialized as 0.5) is the location parameter and 0.1 is the scale. The value of Fm^g is recalculated at the end of each generation according to the method for CRm^g. Moreover, if CR_i^g or F_i^g is outside [0, 1], they are regenerated.

Although the above parameter adaptation scheme is similar to SHADE [39] and SaDE [20], it differs from them. Taking CR as the example, UMDE updates CRm^g according to each weighted mean W_{CR}^g of the successful CR in the previous LG generations, and each W_{CR}^g has its own weight that calculated by the number of successful CR divided by the total number of successful CR during the LG generations. However, SHADE updates CRm^g by randomly selecting a value from the

historical memory which stores a fixed number of the weighted mean of the successful *CR*. SaDE updates CRm^g by calculating the median value of all successful *CR* in the previous *LG* generations.

E. Runtime Complexity of UMDE

According to the pseudocode given in the supplementary material, the runtime complexity of UMDE is analyzed as follow. For the abstract convexity underestimation-based evolutionary stage estimation, the runtime complexity mainly depends on the distance measure, which requires $O(NP^2 \cdot D)$ runtime as Euclidean distance used in UMDE. For the centroid-based strategies, the runtime complexity is determined by the sorting procedure, which is $O(NP \cdot \log_2 NP)$ since the heap sort algorithm is employed in UMDE. For the procedure of strategy selection and parameter adaption, their runtime complexity is relatively small and can be ignored compared with that of the above operations. Hence, the total runtime complexity of UMDE is $O(\max(NP^2 \cdot D \cdot G_{\max}, NP \cdot \log_2 NP \cdot G_{\max}))$ over $G_{\max}$ generations, that is $O(NP^2 \cdot D \cdot G_{\max})$.

From the above analysis, we can know that the runtime complexity of UMDE is mainly determined by the calculation of the average *UE* because we need to compute the distance from each individual to the trial. However, it should be noted that the average *UE* does not need to be calculated when the evolutionary process enters the last stage. Hence, the runtime complexity of UMDE is lower than $O(NP^2 \cdot D \cdot G_{\max})$. Moreover, for expensive-to-evaluate problems (i.e., CEC benchmark problems), the computational burden of an algorithm mainly depends on the function evaluation (FE) [29], [34]. The studies in [29] and [34] show that the computational cost required by calculating the pairwise distance between individuals is comparatively small for computationally expensive problems. Hence, the additional computational overhead cost by UMDE is relatively small when the FE is expensive. The comparison of the computational time between the basic DE and UMDE will be shown in Section V-D.

V. Experimental Results

A. Benchmark Functions and Parameter Settings

In order to demonstrate the performance of UMDE, 23 classical benchmark functions chosen from [12] and [32] are employed in our experiments. In these functions, functions $f_1 - f_{10}$ are unimodal. Functions $f_{11} - f_{23}$ are multimodal. All these functions used in this paper are problems to be minimized. A brief introduction of these functions is given in Table S-I (see supplementary material).

The population size *NP* of UMDE is set to 50. The number *n* of individuals applied to calculate the supporting vector for each trial individual is set to 2. The slope control factor *M* is set to 10 000. The stage control parameter μ is set as 0.85. For the centroid-based strategy, the centroid control parameter *N* is set to *NP*/5. The initial values of *CRm* and *Fm* in the parameter adaptation scheme are both set to 0.5. For each algorithm, each function is independently optimized 30 times.

TABLE I
Results of AR on 30-D Benchmark Functions

UMDE v.s.	SaDE	JADE	EPSDE	SHADE	CoDE	DELU
AR<1	6	3	5	3	0	5
AR=1	0	1	0	1	0	1
AR>1	14	18	15	18	21	16
NA	3	1	3	1	2	1

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

In this part, UMDE is compared with six state-of-the-art DE approaches, i.e., SaDE [5], JADE [13], EPSDE [23], SHADE [39], CoDE [21], and DELU [35]. The parameters of them keep the same as their original papers.

1) Convergence Speed and Successful Rate: In this experiment, a threshold value is defined to compare the convergence speed of different approaches. All algorithms are terminated when the function error is lower than the given threshold value, or the number of FEs reaches the maximum number of function evaluations (MaxFEs). The function error is defined as $(f(x) - f(x^*))$, where x and x^* represent the best solution found so far and the global optimum of the problem, respectively. The mean number FEs (Mean FEs) required to reach the threshold and the successful rate (SR) are recorded. In addition, another criterion, i.e., acceleration rate (AR) is selected from [40] to compare the convergence speed between UMDE and the competitors. AR is calculated by the FEs of other algorithms divided by that of UMDE, where AR $<$ 1, AR $=$ 1, and AR $>$ 1 indicate that the convergence speed of UMDE is slower, almost the same as, and faster than that of the competitor, respectively. In this experiment, the threshold value is set to $1.00E-05$, and MaxFEs is set to be 300 000.

Table S-II (see supplementary material) summarizes the results of Mean FEs, SR, and AR, where "NA" indicates that the corresponding algorithm cannot find solutions below the threshold within the given MaxFEs for any runs. For clarity, the best results are highlighted in boldface. The results of AR reported in Table I indicate that SaDE, JADE, EPSDE, SHADE, CoDE, and DELU converge faster than UMDE on 6, 3, 5, 3, 0, and 5 functions, respectively. UMDE shows faster convergence than SaDE, JADE, EPSDE, SHADE, CoDE, and DELU on 14, 18, 15, 18, 21, and 16 out of 23 functions, respectively. SaDE, EPSDE, and CoDE fail to converge to the threshold on 3, 3, and 2 functions, respectively. JADE, CoDE, and DELU cannot reach the threshold on one function. UMDE successfully find the solution below the threshold on all functions.

Table II reports the total average results of Mean FEs and SR, where NA is replaced by the value of MaxFEs in the calculation. As shown in Table II, the proposed UMDE requires the least FEs (i.e., $5.03E+04$) to reach the threshold. The FEs cost by SaDE, JADE, EPSDE, SHADE, CoDE, and DELU are $7.53E+04$, $7.34E+04$, $7.88E+04$, $6.74E+04$, $9.85E+04$, and $5.65E+04$, respectively. Namely, UMDE saves 33.18%, 31.39%, 36.12%, 25.36%, 48.90%, and 12.21% FEs, respectively, compared to SaDE, JADE, EPSDE, SHADE, CoDE, and DELU. Moreover, UMDE achieves the highest SR (i.e., 0.988) as it reaches the threshold value with 100% SR on 21 out of 23 functions. The SR obtained by SaDE, JADE,

TABLE II
TOTAL AVERAGE MEAN FES AND SR FOR 30-D BENCHMARK FUNCTIONS

	SaDE	JADE	EPSDE	SHADE	CoDE	DELU	UMDE
FEs	7.53E+04	7.34E+04	7.88E+04	6.74E+04	9.85E+04	5.65E+04	5.03E+04
SR	0.846	0.953	0.855	0.954	0.912	0.955	0.988

TABLE III
RESULTS OF WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND ITS COMPETITORS ON 30-D BENCHMARK FUNCTIONS

UMDE v.s.	SaDE	JADE	EPSDE	SHADE	CoDE	DELU
w	15	16	13	18	19	13
t	5	4	8	3	2	6
l	3	3	2	2	2	4

EPSDE, SHADE, CoDE, and DELU are 0.846, 0.953, 0.855, 0.954, 0.912, and 0.955, respectively, since they fail to reach the threshold on some functions.

Fig. S-3 (see supplementary material) presents the convergence curves of the seven DE variants on six representative functions. Clearly, UMDE always converges with three stages in the curve because it uses different strategies in different stages that determined by the abstract convex underestimation-based evolutionary stage estimation scheme. At the beginning of UMDE, it converges slower than other algorithms due to the basic rand-based strategies used in the first stage. However, at the second stage, UMDE gradually converges faster than the competitors as a result of the centroid-based strategies. At the last stage, UMDE quickly converges to the global optimum since it uses the best-so-far solution-based strategies.

2) Final Solution's Quality: The mean and standard deviation values of the function error provided by each algorithm within the given MaxFEs are used to evaluate the final solution's quality. In addition, the Wilcoxon signed-rank test [41] at a 0.05 significance level (α) is utilized to judge the significant difference between any two approaches. For all algorithms, MaxFES is set to $2000 \times D$ in this experiment.

Table S-III (see supplementary material) presents the mean and standard deviation values of the 30-D functions, where the "+," "≈," and "−" represent that the result of UMDE is significantly better than, almost similar to, and significantly worse than that of the compared algorithm, respectively. Based on the data summarized in Table S-III, it is obvious that UMDE provides better results than its competitors for most of the benchmark functions. Furthermore, the results obtained by Wilcoxon signed-rank test are given in Table III, where w, t, and l indicate that UMDE, respectively, wins on w functions, ties on t functions, and loses on l functions, compared to its competitors. As shown in Table III, UMDE is significantly outperforms SaDE, JADE, EPSDE, SHADE, CoDE, and DELU on 15, 16, 13, 18, 19, and 13 out of 23 functions, respectively. SaDE, JADE, EPSDE, SHADE, CoDE, and DELU are significantly superior to UMDE only on 3, 3, 2, 2, 2, and 4 functions, respectively.

3) Multiple Problem Analysis: In Table S-III, we only perform the single problem analysis according to the Wilcoxon signed-rank test. However, the multiple problem analysis [42], which allows us to simultaneously compare two or more algorithms over a set of problems, is also important. Here, the multiple problem Wilcoxon signed-rank test at $\alpha = 0.05$ is utilized to compare the performance of each algorithm.

TABLE IV
MULTIPLE PROBLEM ANALYSIS BY THE WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND ITS COMPETITORS

UMDE v.s.	SaDE	JADE	EPSDE	SHADE	CoDE	DELU
R^+	151	153	161	193	193	111
R^-	39	57	29	38	38	60
p-value	0.0242	0.0731	0.0079	0.0071	0.0071	0.2668
Significant	+	≈	+	+	+	≈

TABLE V
RESULTS OF WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND ITS COMPETITORS ON 50-D AND 100-D BENCHMARK FUNCTIONS

UMDE v.s.		SaDE	JADE	EPSDE	SHADE	CoDE	DELU
D=50	w	14	14	16	12	15	12
	t	7	5	5	6	6	7
	l	2	4	2	5	2	4
D=100	w	16	16	21	15	19	13
	t	3	3	1	2	1	5
	l	4	4	1	6	3	5

Table IV presents the results of multiple problem analysis obtained by Wilcoxon signed-rank test, where R^- and R^+ represent the sum ranks that UMDE is worse than and better than the compared algorithm, respectively. As seen, UMDE achieves higher R^+ values than R^- compared with its competitors. Moreover, UMDE exhibits significantly better performance than SaDE, EPSDE, SHADE, and CoDE since their p-values are less than 0.05. Although the p-value indicate that the performance of UMDE is almost similar to those of JADE and DELU, UMDE is significantly better than JADE and DELU on the majority of functions based on the single problem analysis reported in Table III.

C. Scalability of UMDE

In order to study the relationship between the performance of UMDE and the dimension of the problem, a scalability test is, respectively, conducted on 50-D and 100-D functions. In this experiment, we further compare UMDE with the six DE approaches used in Section V-B. Tables S-IV and S-V (see supplementary material) summarize the results of 50-D and 100-D functions that obtained within $3000 \times D$ FEs, respectively. From the data, we can see that the performance of UMDE is not affected when the complexity of the problem increases with the growth of the dimension. UMDE is still significantly superior to its competitors on the majority of problems. Additionally, Table V lists the results achieved by Wilcoxon signed-rank test. For 50-D functions, UMDE significantly outperforms SaDE, JADE, EPSDE, SHADE, CoDE, and DELU on 14, 14, 16, 12, 15, and 12 out of 23 functions, respectively. SaDE, JADE, EPSDE, SHADE, CoDE, and DELU provide significantly better results than UMDE on 2, 4, 2, 5, 2, and 4 functions, respectively. For 100-D functions, UMDE obtains significantly better results than SaDE, JADE, EPSDE, SHADE, CoDE, and DELU on 16, 16, 21, 15, 19, and 13 out of 23 functions, respectively. SaDE, JADE, EPSDE, SHADE, CoDE, and DELU get significantly better performance than UMDE only on 4, 4, 1, 6, 3, and 5 functions, respectively.

TABLE VI
AVERAGE RANKINGS OF UMDE AND ITS COMPETITORS ON 50-D AND 100-D BENCHMARK FUNCTIONS (FRIEDMAN)

		SaDE	JADE	EPSDE	SHADE	CoDE	DELU	UMDE
D=50	Ranking	4.98	3.80	4.87	3.09	5.24	3.41	**2.61**
D=100	Ranking	5.24	3.54	5.43	3.67	5.13	2.72	**2.26**

TABLE VII
RESULTS OF WILCOXON SIGNED-RANK AND FRIEDMAN TEST BETWEEN UMDE AND UMDE-3FS, UMDE-2FS, AND UMDE-2US

	UMDE-3fs	UMDE-2fs	UMDE-2us	UMDE
w	17	18	17	-
t	4	3	5	-
l	2	2	1	-
Ranking	2.83	2.72	3.04	1.41

Table VI shows the average rankings of 50-D and 100-D functions obtained by Friedman test [43]. The best ranking is marked in boldface. Based on the average ranking given in Table VI, it is obviously observed that UMDE gets the first ranking, followed by SHADE, DELU, JADE, EPSDE, SaDE, and CoDE for $D = 50$. For $D = 100$, the best ranking is also achieved by UMDE. DELU is the runner up, and follwed by JADE, SHADE, CoDE, SaDE, and EPSDE. The above analysis shows that the performance of UMDE is not affected by the increase of the dimension of the problem.

D. Effects of UMDE Components

Four components of UMDE are the abstract convex underestimation-based stage estimation, the centroid-based mutation strategy, the stage strategy adaptation, and the parameter adaptation. In this section, various experiments are conducted on the 30-D functions to identify the effect of each component. The mean and standard deviation values obtained within $2000 \times D$ FEs are recorded.

1) Abstract Convex Underestimation-Based Stage Estimation: In order to investigate the effect of the abstract convex underestimation-based evolutionary stage estimation, UMDE with three fixed stages (UMDE-3fs), UMDE with two fixed stages (UMDE-2fs), and UMDE with two abstract convex underestimation-based stages (UMDE-2us) are compared with UMDE. In UMDE-3fs and UMDE-2fs, the evolutionary process is, respectively, divided into three and two equally size stages according to the MaxFEs, while UMDE-2us includes two abstract convex underestimation-based stages (S_1 and S_2). To be fair, the same parameter settings are utilized in these four algorithms. The results are summarized in Table S-VI (see supplementary material). As seen, UMDE significantly outperforms the other three UMDE variants on most of the functions. Meanwhile, from the Wilcoxon signed-rank and Friedman test results listed in Table VII, we can know that UMDE is significantly superior to the UMDE variant with fixed stages or two underestimation-based stages. In addition, to investigate the behavior of the abstract convex underestimation, six functions are selected to depict the curve of *UE*. As shown in Fig. S-4 (see supplementary material), the *UE* decreases with the evolutionary process.

2) Centroid-Based Mutation Strategy: To demonstrate the effect of the centroid-based mutation strategy, UMDE using only centroid-based strategies (strategies used in S_2), UMDE using only rand-based strategies (strategies used in S_1), and UMDE using only best-so-far solution-based strategies (strategies used in S_3), are employed in this experiment. They are denoted as UMDE-centroid, UMDE-rand, and UMDE-best, respectively. The same parameter setting of $CR = 0.5$, $F = 0.5$, and $NP = 50$ are used in these three algorithms, and the mutation strategy is randomly chosen from the strategy pool. Table S-VII (see supplementary material) shows the results obtained by these three UMDE variants. It is clear that UMDE-centroid gets better results on most of the functions compared to UMDE-rand and UMDE-best. Furthermore, according to the results of Wilcoxon signed-rank and Friedman test given in Table VIII, UMDE-centroid performs significantly better than the other two UMDE variants.

TABLE VIII
RESULTS OF WILCOXON SIGNED-RANK AND FRIEDMAN TEST BETWEEN UMDE-CENTROID AND UMDE-RAND, UMDE-BEST

	UMDE-rand	UMDE-best	UMDE-centroid
w	18	15	-
t	3	2	-
l	2	6	-
Ranking	2.61	1.89	1.50

TABLE IX
AVERAGE RANKINGS OF DE WITH DIFFERENT STRATEGIES (FRIEDMAN)

DE/	best/2	lbest/1	current-to-best/1	current-to-$p$best/1
Ranking	3.50	2.83	6.304	3.26
DE/	centroid/2	current-to-centroid/1	rand-to-best/1	rand-to-centroid/1
Ranking	**2.41**	6.22	6.17	5.30

In addition, we also compare DE using the proposed centroid-based mutation strategies with DE using DE/best/2, DE/lbest/1 [12], DE/current-to-best/1, DE/current-to-$p$best/1 with archive [13], and DE/rand-to-best/1 to demonstrate the effectiveness of the centroid-based mutation strategies. The same parameter setting of $CR = 0.5$, $F = 0.5$, and $NP = 50$ are used for them. The results are listed in Table S-VIII (see supplementary material). As seen, DE/centroid/2 outperforms its competitors on the majority of problems. Moreover, DE/centroid/2 gets the first ranking according to the Friedman test results summarized in Table IX. DE/current-to-$p$best/1 is superior to DE/current-to-centroid/1, while DE/current-to-centroid/1 performs better than DE/current-to-best/1. DE/rand-to-centroid/1 obtains obviously better performance than its similar strategy DE/current-to-best/1.

3) Stage Strategy Adaptation: In this experiment, UMDE is compared with UMDE without stage strategy adaptation (UMDE-wosa), UMDE using DE/rand/1, DE/centroid/2, and DE/best/2 (UMDE-111), UMDE using DE/rand/1, DE/current-to-centroid/1, and DE/rand-to-best/1 (UMDE-123), and UMDE using DE/current-to-rand/1, DE/current-to-centroid/1, and DE/best/2 (UMDE-321) to study the effect of the stage strategy adaptation. In UMDE-wosa, one of the strategies in the strategy pool is randomly chosen to generate a mutant vector, while the fixed strategy is used in different stage of UMDE-111, UMDE-123, and UMDE-321. The parameter settings of these four UMDE variants are the same as UMDE in Section V-A. Table S-IX (see supplementary

TABLE X
RESULTS OF WILCOXON SIGNED-RANK AND FRIEDMAN TEST BETWEEN UMDE AND UMDE-WOSA, UMDE-111, UMDE-123, AND UMDE-321

	UMDE-wosa	UMDE-111	UMDE-123	UMDE-321	UMDE
w	15	10	22	21	-
t	5	9	1	1	-
l	3	4	0	1	-
Ranking	2.61	2.24	4.65	3.87	1.63

TABLE XI
RESULTS OF WILCOXON SIGNED-RANK AND FRIEDMAN TEST BETWEEN UMDE AND UMDE($CR = 0.1$), UMDE($CR = 0.5$), AND UMDE($CR = 0.9$)

	UMDE(CR=0.1)	UMDE(CR=0.5)	UMDE(CR=0.9)	UMDE
w	20	20	21	-
t	3	2	0	-
l	0	1	2	-
Ranking	2.91	2.33	3.52	1.24

TABLE XII
RESULTS OF WILCOXON SIGNED-RANK TEST AND FRIEDMAN TEST BETWEEN UMDE AND UMSADE, UMJDE, UMJADE, AND UMSHADE

	UMSaDE	UMjDE	UMJADE	UMSHADE	UMDE
w	15	18	19	12	-
t	5	2	3	7	-
l	3	3	1	4	-
Ranking	3.20	3.67	4.17	2.15	1.80

TABLE XIII
RESULTS OF KRUSKAL-WALLIS AND FRIEDMAN TEST FOR DIFFERENT M

	M=4000	M=6000	M=8000	M=10000	M=12000	M=14000
K^+	4	3	7	15	11	4
$K^{\approx}$	1	4	5	7	2	4
K^-	18	16	11	1	10	15
Ranking	4.87	4.67	2.85	1.98	2.50	4.13

material) summarizes the mean and standard deviation values of these five UMDE variants. Clearly, UMDE gets significantly better results than the other four UMDE variants on most of the functions. Additionally, the results achieved by Wilcoxon signed-rank and Friedman test are shown in Table X. Obviously, UMDE is the best algorithm among these five UMDE variants.

4) Parameter Adaptation: In order to study the effect of the parameter adaptation, three UMDE variants using different settings of F and CR are compared with UMDE. The parameter F of them is set to 0.5 as suggested in [1] and [12], and the parameter CR is set to 0.1, 0.5, and 0.9, respectively. Therefore, they are denoted as UMDE($CR = 0.1$), UMDE($CR = 0.5$), and UMDE($CR = 0.9$), respectively. The experimental results obtained by them are listed in Table S-X (see supplementary material). It can be observed that UMDE significantly outperforms the other three UMDE variants without parameter adaptation. Meanwhile, according to the results shown in Table XI, UMDE is the most effective one among the four UMDE algorithms.

To further demonstrate the superior performance of the parameter adaptation scheme, four UMDE using the parameter adaptation techniques presented in SaDE [20], jDE [44], JADE [13], and SHADE [39] (denoted as UMSaDE, UMjDE, UMJADE, and UMSHADE, respectively) are compared with UMDE. The parameter settings of their parameter adaptation schemes keep the same as their original papers, and the other parameters are the same as UMDE. The mean and standard deviation values are reported in Table S-XI (see supplementary material). As seen, UMDE is significantly superior to the competitors on the majority of functions. Moreover, according to the results of Wilcoxon signed-rank test summarized in Table XII, UMDE significantly outperforms UMSaDE, UMjDE, UMJADE, and UMSHADE on 15, 18, 19, and 12 out of 23 functions, respectively. Additionally, the results of Friedman test reveal that UMDE obtains the best ranking, followed by UMSHADE, UMSaDE, UMjDE, and UMJADE.

E. Parameter Study of UMDE

In the proposed UMDE, four parameters, i.e., n, M, μ, and N need to be optimized. As we have studied the influence of the parameter n on the algorithm and concluded that $n = 2$ is more preferable in [35], here, we just investigate the influence of the other three parameters. In the following experiments, the Friedman and Kruskal-Wallis test with multiple comparisons [45] are utilized to assess the impact of each parameter. The MaxFEs is set to be $2000 \times D$ and the results are averaged.

1) Slope Control Factor (M): As a suitable M can obtain a good underestimation with small error, the impact of M needs to be investigated. For this purpose, experiments of UMDE with six different M (varies from 4000 to 14 000 with a step of 2000) are conducted on the 30-D functions. Table S-XII (see supplementary material) summarizes the mean and standard deviation values. Moreover, the results obtained by Kruskal-Wallis and Friedman test are listed in Table XIII, where K^+ indicates that the algorithm is the best among all compared algorithms, and K^- and $K^{\approx}$ denote that the algorithm is significantly worse than and almost similar to the best algorithm, respectively. The results of Kruskal-Wallis test reveal that $M = 10\,000$ achieves the best performance as it finds the best results on 15 out of 23 functions. Meanwhile, $M = 10\,000$ gets the best ranking according to the results of Friedman test. In order to further demonstrate the performance of $M = 10\,000$, the curves of UE in a single run for some representative functions are depicted in Fig. S-5. As seen, $M = 10\,000$ provides the lower UE compare to the other M. The above results and analysis indicate that $M = 10\,000$ is more suitable for UMDE.

2) Stage Control Parameter (μ): In this experiment, UMDE with nine different stage control parameters μ (varies from 0.55 to 0.95 with a step of 0.05) are employed to study the influence of μ over the 30-D functions. The mean and standard deviation values are summarized in Tables S-XIII and S-XIV (see supplementary material). Additionally, the results of Kruskal-Wallis and Friedman test are given in Table XIV. The results of Kruskal-Wallis test reveal that $\mu = 0.85$ and $\mu = 0.9$ all provide the best result on 13 out of 23 problems. However, $\mu = 0.85$ is obviously worse than the best result on 5 functions, while $\mu = 0.9$ performs significantly worse than the best result on 7 functions. Meanwhile, $\mu = 0.85$ obtains the first ranking according to the results of Friedman test. In addition, Fig. S-6 (see supplementary material) shows the changes of the mean FEs and SR with

TABLE XIV
RESULTS OF KRUSKAL-WALLIS AND FRIEDMAN TEST FOR DIFFERENT μ

	μ=0.55	μ=0.6	μ=0.65	μ=0.7	μ=0.75	μ=0.8	μ=0.85	μ=0.9	μ=0.95
K^+	5	4	4	4	7	11	13	13	8
$K^{\approx}$	1	2	3	4	3	3	5	3	2
K^-	17	17	16	15	13	9	5	7	13
Ranking	6.74	6.48	6.00	5.30	4.98	3.65	3.22	3.63	5.00

TABLE XV
RESULTS OF KRUSKAL-WALLIS AND FRIEDMAN TEST FOR DIFFERENT N

	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 25$
K^+	8	15	15	8	8
$K^{\approx}$	2	7	6	4	2
K^-	13	1	2	11	13
Ranking	3.61	2.04	2.15	3.39	3.80

the increasing of μ for some selected functions. It also shows that $\mu = 0.85$ is the most choice for the proposed UMDE.

3) Centroid Control Parameter (N): To investigate the impact of N, UMDE with five different N, i.e., 5, 10, 15, 20, and 25, are conducted on the 30-D benchmark functions. The results of mean and standard deviation are listed in Table S-XV (see supplementary material). Meanwhile, the results obtained by Kruskal-Wallis and Friedman test are listed in Table XV. From the results of Kruskal-Wallis test, it can be observed that $N = 10$ and $N = 15$ provide better results as they all obtain the best results on 15 functions. However, the results of Friedman test reveal that $N = 10$ is more suitable since it obtains the best ranking. Moreover, the box plots of FEs that required to reach the predefined threshold ($1.00E-05$) with different N on some representative functions are presented in Fig. S-7 (see supplementary material). It also indicates that the value of N could be selected within the range of [10, 15]. Therefore, we can conclude that $N = 10$ is more preferable based on the above results and analysis.

F. Computational Cost of UMDE

As discussed in Section IV-F, the runtime complexity of UMDE is mostly determined by calculating the distance from the trial individual to each individual. Here, we compare the runtime of UMDE and the basic DE (rand/1/bin) with $F = 0.9$ and $CR = 0.95$ [46], [47] on the functions listed in Table S-I and some selected CEC 2014 functions (f_2, f_3, f_7, and f_8) at $D = 30$ and $D = 50$. The stopping rule is the function error less than the threshold value (0.00001) or FEs reaches 300 000. The runtime ratio $\text{Cost}_{\text{UMDE}}/\text{Cost}_{\text{DE}}$ is computed, where $\text{Cost}_{\text{UMDE}}$ and Cost_{DE} represent the average runtime of UMDE and DE over 30 independent runs, respectively.

From the previous analysis, we have demonstrated that UMDE requires less FEs to achieve the predefined threshold value. The runtime ratios of the 23 benchmark functions at 30-D and 50-D are approximately equal to 3.833 and 3.697, respectively. For the selected CEC 2014 functions at 30-D and 50-D, the runtime ratios are 3.145 and 3.287, respectively. The above results indicate that UMDE is acceptable for the practical problems, especially, in the case of problems with expensive FE.

TABLE XVI
RESULTS OF WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND THE NON-DE COMPETITORS ON 30-D AND 50-D BENCHMARK FUNCTIONS

		CLPSO	CMA-ES	GL-25			CLPSO	CMA-ES	GL-25
D=30	w	22	16	22	D=50	w	21	15	22
	t	1	2	1		t	1	3	1
	l	0	5	0		l	1	5	0

G. Comparison of UMDE With Non-DE Algorithms

In this part, we compare UMDE with three advanced non-DE EAs: CLPSO [48], CMA-ES [49], and GL-25 [50]. The parameters of these three algorithms are set according to their original papers. The stopping criteria for $D = 30$ and $D = 50$ problems are the number of FES reaches $2000 \times D$ and $3000 \times D$ MaxFEs, respectively.

Tables S-XVI and S-XVII (see supplementary material) summarize the mean and standard deviation values provided by the three compared non-DE algorithms and UMDE for 30-D and 50-D functions, respectively. Obviously, UMDE is significantly superior to its competitors on the majority of functions. Additionally, the results obtained by Wilcoxon signed-rank test are given in Table XVI. For $D = 30$, UMDE significantly outperforms CLPSO, CMA-ES, and GL-25 on 22, 16, and 22 out of 23 functions, respectively. CLPSO and GL-25 is not significantly better than UMDE on any functions. CMA-ES observably outperforms UMDE on five functions. For $D = 50$, UMDE gets significantly better performance than CLPSO, CMA-ES, and GL-25 on 21, 15, and 22 out of 23 functions, respectively. CLPSO performs significantly better than UMDE only on one function. CMA-ES significantly outperforms UMDE on five functions. GL-25 is not significantly superior to UMDE on any problems.

H. Performance on CEC Benchmark Set

In this section, the performance of UMDE is further demonstrated over the CEC 2013 [51] and CEC 2014 benchmark sets [52]. Detailed introduction of the function can be found in [51] and [52]. In the following experiments, all compared algorithms are terminated when FEs reaches the MaxFEs. As Liang *et al.* [51] suggested, the MaxFEs is set to $10\,000\times D$.

We first compare UMDE with four DE algorithms (i.e., SHADE [39], ZEPDE [14], IDE [38], and SinDE [53]) on 30-D and 50-D CEC 2013 benchmark functions. The parameter settings for these four approaches are the same as in their original literatures. Tables S-XVIII and S-XIX (see supplementary material) report the mean and standard deviation values of 30-D and 50-D functions, respectively. It is clear that UMDE provides better performance than its competitors on the majority of functions. Additionally, Table XVII reports the results provided by Wilcoxon signed-rank test. For 30-D functions, the proposed UMDE obtains significantly better results than SHADE, ZEPDE, IDE, and SinDE on 13, 19, 16, and 21 out of 28 functions, respectively. SHADE, ZEPDE, IDE, and SinDE are significantly superior to UMDE on 8, 3, 7, and 6 functions, respectively. For 50-D functions, UMDE significantly outperforms SHADE, ZEPDE, IDE, and SinDE on 16, 17, 16, and 17 out of 28 functions, respectively. However, UMDE is significantly worse than SHADE, ZEPDE, IDE, and SinDE on 7, 6, 7, and 7 functions, respectively.

TABLE XVII
RESULTS OF WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND ITS COMPETITORS ON 30-D AND 50-D CEC 2013 FUNCTIONS

		SHADE	ZEPDE	IDE	SinDE			SHADE	ZEPDE	IDE	SinDE
	w	13	19	16	21		w	16	17	16	17
D=30	t	7	6	5	1	D=50	t	5	5	5	4
	l	8	3	7	6		l	7	6	7	7

TABLE XVIII
RESULTS OF WILCOXON SIGNED-RANK TEST BETWEEN UMDE AND ITS COMPETITORS ON 10-D, 30-D, AND 50-D CEC 2014 FUNCTIONS

UMDE v.s.	D=10			D=30			D=50		
	w	t	l	w	t	l	w	t	l
L-SHADE	16	9	5	16	13	1	17	8	5
LSHADE-ND	16	11	3	15	13	2	17	9	4
LSHADE-EpSin	14	7	9	11	10	9	13	5	12
MC-SHADE	20	9	1	21	7	2	22	5	3
iLSHADE	13	10	7	8	16	6	11	13	6

In addition, UMDE is also compared with the top DE methods in CEC 2014-2016 competition winners, i.e., L-SHADE [54], LSHADE-ND [55], LSHADE-EpSin [56], MC-SHADE [57], and iLSHADE [58], on 10-D, 30-D, and 50-D CEC 2014 functions as suggested in [59]. All parameters of them are determined based on their original papers. The results of 10-D, 30-D, and 50-D functions are given in Tables S-XX–S-XXII (see supplementary material). Moreover, the Wilcoxon signed-rank test results listed in Table XVIII reveal that UMDE is significantly superior to L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE on 16, 16, 14, 20, and 13 out of 30 functions for 10-D functions, respectively. L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE significantly outperform UMDE on 5, 3, 9, 1, and 7 functions, respectively. For 30-D functions, UMDE is significantly better than L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE on 16, 15, 11, 21, and 8 functions, respectively. L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE perform significantly better than UMDE on 1, 2, 9, 2, and 6 functions, respectively. For 50-D functions, L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE provides significantly better results than UMDE on 5, 4, 12, 3, and 6 functions, respectively. However, UMDE significantly outperforms L-SHADE, LSHADE-ND, LSHADE-EpSin, MC-SHADE, and iLSHADE on 17, 17, 13, 22, and 11 out of 30 functions, respectively.

VI. CONCLUSION

In this paper, an abstract convex underestimation assisted multistage DE (UMDE) is presented to enhance the performance of DE. In UMDE, the mutation strategy is dynamically selected according to the evolutionary stages estimated by the variation of the value of *UE*, and three new centroid-based strategies are designed to balance the population diversity and fast convergence of the algorithm. In addition, a simple parameter adaption scheme is applied to adaptively select parameters *F* and *CR* during the evolution. We have verified the performance of UMDE by comparing with some advanced DE approaches and non-DE EAs over a suite of classical benchmark problems, CEC 2013, and CEC 2014 benchmark sets. The experimental results show that UMDE outperforms, or at least comparably to its competitors. Additionally, the effect of each components and the sensitivity of parameters in UMDE have been experimentally studied. In the future, we will investigate the stage control parameter adaption strategy and apply the proposed algorithm to real-world problems.

REFERENCES

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

[2] A. Bhattacharya and P. K. Chattopadhyay, "Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch," *IEEE Trans. Power Syst.*, vol. 25, no. 4, pp. 1955–1964, Nov. 2010.

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

[4] G. Zhang, X.-G. Zhou, X.-F. Yu, X.-H. Hao, and L. Yu, "Enhancing protein conformational space sampling using distance profile-guided differential evolution," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, to be published, doi: 10.1109/TCBB.2016.2566617.

[5] D. Qiao and G. K. H. Pang, "A modified differential evolution with heuristic algorithm for nonconvex optimization on sensor network localization," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1676–1689, Mar. 2016.

[6] 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.

[7] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.

[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1996.

[9] I. Rechenberg, *Evolutions Strategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*. Stuttgart, Germany: Fromman-Holzboog, 1973.

[10] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 561–578, Oct. 2007.

[11] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2066–2081, Dec. 2013.

[12] W.-J. Yu *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, Jul. 2014.

[13] 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.

[14] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 219–232, Jan. 2016.

[15] M.-Y. Cheng and D.-H. Tran, "Two-phase differential evolution for the multiobjective optimization of time–cost tradeoffs in resource-constrained construction projects," *IEEE Trans. Eng. Manag.*, vol. 61, no. 3, pp. 450–461, Aug. 2014.

[16] A. M. Rubinov, *Abstract Convexity and Global Optimization, of Nonconvex Optimization and Its Applications*. Dordrecht, The Netherlands: Kluwer Academic, 2000.

[17] G. Beliakov, "Geometry and combinatorics of the cutting angle method," *Optimization*, vol. 52, nos. 4–5, pp. 379–394, Jul. 2003.

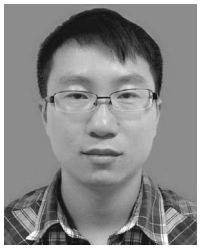
[18] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1970.

[19] G. Beliakov, "Extended cutting angle method of global optimization," *Pac. J. Optim.*, vol. 4, no. 1, pp. 153–176, 2008.

[20] 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.

[21] 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.

[22] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.
[23] 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.
[24] W. Gong, A. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: An empirical study," *Inf. Sci.*, vol. 181, no. 24, pp. 5364–5386, Dec. 2011.
[25] Q.-K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adapting strategy and control parameters," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 394–408, Jan. 2011.
[26] W. Gong, A. Zhou, and Z. Cai, "A multioperator search strategy based on cheap surrogate models for evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 746–758, Oct. 2015.
[27] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 1041–1048.
[28] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, no. 2, pp. 329–345, Feb. 2016.
[29] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013.
[30] Y. Zhou, X. Li, and L. Gao, "A differential evolution algorithm with intersect mutation operator," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 390–401, Jan. 2013.
[31] S.-M. Guo, C.-C. Yang, P.-H. Hsu, and J. S.-H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 717–730, Oct. 2015.
[32] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 634–647, Apr. 2013.
[33] 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.
[34] 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.
[35] X.-G. Zhou, G.-J. Zhang, X.-H. Hao, and L. Yu, "A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization," *Comput. Oper. Res.*, vol. 75, no. 11, pp. 132–149, Nov. 2016.
[36] G. Liu, C. Xiong, and Z. Guo, "Enhanced differential evolution using random-based sampling and neighborhood mutation," *Soft Comput.*, vol. 19, no. 8, pp. 2173–2192, Aug. 2015.
[37] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb. 2008.
[38] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.
[39] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 71–78.
[40] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.
[41] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Hoboken, NJ, USA: Wiley, 2009.
[42] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.
[43] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.
[44] 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.
[45] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *J. Amer. Stat. Assoc.*, vol. 47, no. 260, pp. 583–621, Dec. 1952.
[46] R. Storn and K. 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.
[47] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *IEEE Trans. Ind. Inf.*, vol. 9, no. 1, pp. 89–99, Feb. 2013.
[48] 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.
[49] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.
[50] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1088–1113, Mar. 2008.
[51] 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, 2013.
[52] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Tech. Rep. 201311, Dec. 2013.
[53] D. Amer, S. Bouzoubia, and I. Boukhalfa, "A sinusoidal differential evolution algorithm for numerical optimisation," *Appl. Soft Comput.*, vol. 27, pp. 99–126, Feb. 2015.
[54] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 1658–1665.
[55] K. M. Sallam, R. A. Sarker, D. L. Essam, and S. M. Elsayed, "Neurodynamic differential evolution algorithm and solving CEC2015 competition problems," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 1033–1040.
[56] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "An ensemble sinusoidal parameter adaptation incorporated with L-shade for solving CEC2014 benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 2958–2965.
[57] A. Viktorin, M. Pluhacek, and R. Senkerik, "Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 4797–4803.
[58] J. Brest, M. S. Maučec, and B. Bošković, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 1188–1195.
[59] A. Zamuda and J. Brest, "Self-adaptive control parameters' randomization frequency and propagations in differential evolution," *Swarm Evol. Comput.*, vol. 25, pp. 72–99, Dec. 2015.



Xiao-Gen Zhou is currently pursuing the Ph.D. degree in control science and engineering from the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China.

His current research interests include intelligent information processing, optimization theory and algorithm design, and bioinformatics.

Gui-Jun Zhang received the Ph.D. degree in control theory and control engineering from Shanghai Jiaotong University, Shanghai, China, in 2004.

He is currently a Professor with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His current research interests include intelligent information processing, optimization theory and algorithm design, and bioinformatics.