Differential Evolution With Underestimation-Based Multimutation Strategy

Xiao-Gen Zhou and Gui-Jun Zhang

***Abstract*—As we know, the performance of differential evolution (DE) highly depends on the mutation strategy. However, it is difficult to choose a suitable mutation strategy for a specific problem or different running stages. This paper proposes an underestimation-based multimutation strategy (UMS) for DE. In the UMS, a set of candidate offsprings are simultaneously generated for each target individual by utilizing multiple mutation strategies. Then a cheap abstract convex underestimation model is built based on some selected individuals to obtain the underestimation value of each candidate offspring. According to the quality of each candidate offspring measured by the underestimation value, the most promising candidate solution is chosen as the offspring. Compared to the existing probability-based multimutation techniques, no mutation strategies are lost during the search process as each mutation strategy has the same probability to generate a candidate solution. Moreover, no extra function evaluations are produced because the candidate solutions are filtered by the underestimation value. The UMS is integrated into some DE variants and compared with their original algorithms and several advanced DE approaches over the CEC 2013 and 2014 benchmark sets. Additionally, a well-known real-world problem is employed to evaluate the performance of the UMS. Experimental results show that the proposed UMS can improve the performance of the advanced DE variants.**

***Index Terms*—Differential evolution (DE), evolutionary algorithm (EA), global optimization, multimutation, underestimation.**

I. Introduction

DIFFERENTIAL evolution (DE), proposed by Storn and Price [1], is a population-based metaheuristic optimization algorithm [2]. It is well-known for its simple structure, ease of use, speed, robustness, and efficiency. During the last few decades, DE has shown its advantages in various numerical optimization problems and practical applications, such as protein structure prediction (PSP) [3]–[5], topological active net [6], sensor network localization [7], and flow shop scheduling [8]. The other applications and improved variants of DE are summarized in [9]–[11].

Manuscript received October 7, 2017; revised December 20, 2017 and January 28, 2018; accepted January 30, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61773346 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: zxg@zjut.edu.cn; 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.2018.2801287

DE evolves the population to the global optimum by the mutation, crossover, and selection operator. The performance of DE is highly influenced by the mutation strategy whose objective is to increase the population diversity and consequently avoid getting trapped into a local optimum [12]. Over the past few decades, a variety of mutation strategies, such as DE/lbest/1 [13], DE/current-to-*p*best/1 [14], and ranking-based mutations [15], [16], have been designed to improve the performance of DE. However, a mutation strategy may be suitable for one problem or a class of problems, but may not work well for other problems because of the different mathematical characteristics of different problems. Therefore, choosing the best one among different mutation strategies is a challenge for a specific problem at hand. On the other hand, as the evolution proceeds, the population may evolve into different stages. Employing different strategy in different stages may be more effective.

During the last decade, various techniques that combined different mutation strategies were proposed to enable DE to solve a wide range of problems and further improve the performance. The most frequently used approach is the probability model-based strategy self-adaptive technique, in which one mutation strategy is chosen according to the probability model built based on the previous successful experience. Qin *et al.* [17] proposed a DE with strategy adaptation (SaDE). For each iteration of SaDE, a mutation strategy is chosen from the strategy candidate pool according to the corresponding posterior probability calculated based on the previous successful experience. Gong *et al.* [18] proposed a strategy adaptation mechanism (SaM) for DE. In SaM, for each target individual, the mutation strategy is chosen from the strategy pool by using a control parameter which is adaptively updated. Fan and Yan [19] proposed a self-adaptive DE with zoning evolution parameters and strategies (ZEPDE). In ZEPDE, each mutation strategy is randomly allocated to the individual. For each mutation strategy, the number of individuals is dynamically updated according to its corresponding selective probability and cumulative probability. Various experiments have verified that the above techniques can effectively improve the performance of DE. However, how to build a proper probability model is still a challenging task [20]. An incorrect model may lead some strategies to lose in a certain running stage even though they are more suitable for other stages.

Instead of selecting the mutation strategy according to a probability model, the multistrategy coexistence technique simultaneously employs multiple strategies for the mutation vector generation. For each target individual, a suite of

candidate offsprings are created by applying multiple different mutation strategies, and the best one is chosen based on the function value. Wang *et al.* [21] proposed a composite DE (CoDE). For each target individual in CoDE, several candidate offspring individuals are simultaneously generated by using multiple diverse mutation strategies and the one with the best function value is selected as the offspring individual. Wang *et al.* [22] proposed a DE with cumulative population distribution information. In this algorithm, two candidate offspring individuals are simultaneously generated for each target individual by using an Eigen coordinate system and the original coordinate system. One of the candidate offspring individuals is selected according to the function value. Mezura-Montes *et al.* [23] introduced a modified DE approach that allows each target individual to generate multiple offsprings by applying different mutation strategies, which combine information of the best individual found so far and also information of the target individual. According to the function value, feasibility, and sum of constraint violation, the best offspring is selected to compete against the corresponding target individual. Although the above algorithms have been proven effective for numerical optimization, it may be unaffordable for expensive-to-evaluate real-world problems as it needs to evaluate multiple offspring individuals in each iteration.

To reduce the computational cost of the multistrategy coexistence approach, various surrogate models [24], [25] are applied to replace the actual function evaluations (FEs). Gong *et al.* [20] introduced a multioperator search strategy for evolutionary algorithms (EAs), in which multiple offspring generation strategies are employed to produce a set of candidate offspring individuals, and only one offspring is survived based on the density value calculated by a cheap surrogate model (CSM). Zhou *et al.* [26] also presented a multioperator method. In this approach, a set of candidate offspring individuals are sampled for each target individual by Gaussian distribution. The quality of each offspring individual is measured by a nonparametric density estimation method and the offspring individual with low quality is filtered. Liu *et al.* [27] proposed a Gaussian process surrogate model assisted EA (GPEME). In GPEME, Sammon mapping is first employed to reduce the dimension of the problem. Then Gaussian process surrogate model is constructed in the low-dimensional space to replace the real FEs. Mallipeddi and Lee [28] introduced an evolving surrogate model-based DE (ESMDE), in which a surrogate model built based on the current population is applied to help DE produce competitive offspring. Experimental results have verified the effectiveness of the above methods. However, it is not trivial to select an appropriate surrogate model for a specific problem because no surrogate model is available to all kinds of problems [29].

This paper proposes an underestimation-based multimutation strategy (UMS) for DE. For each target individual, several candidate offsprings are generated by applying multiple mutation strategies in the UMS. Then a cheap abstract convex underestimation model (CAUM) is constructed based on some selected individuals to filter the candidate offsprings, and one of the candidate offsprings is left as the offspring individual. It should be noted that this paper presents a new strategy to enhance the performance of existing DE algorithms rather than a new DE approach. To this end, the proposed UMS is incorporated into several advanced multimutation-based DE variants. The performance of these UMS-based algorithms is compared with their original DE algorithms, several CEC DE winners, and some state-of-the-art DE algorithms on CEC 2013 and 2014 benchmark sets as well as a real-world problem. Experimental results demonstrate that the UMS-based approaches can achieve better results than the comparison methods on the majority of benchmark problems and the real-world problem.

The reminder of this paper is organized as follows. Section II introduces the original DE and basic underestimation model. The proposed UMS is presented in Section III. Empirical study and analysis are performed in Section IV. Section V draws the conclusions of this paper and discusses the future work.

II. Background

A. Differential Evolution

DE finds the global optimum of a problem by using three simple operators, namely, mutation, crossover, and selection [1], [30]. First, an initial population including NP individuals $\boldsymbol{x}_i^0 = (x_{i,1}^0, x_{i,2}^0, \ldots, x_{i,D}^0)$, $i = 1, 2, \ldots, \mathrm{NP}$ is randomly generated in the search space, where NP and D represent the population size and dimension of the problem, respectively.

Then, in each generation g, each individual $\boldsymbol{x}_i^g$ is regarded as a target individual, and three other different individuals $\boldsymbol{x}_a^g$, $\boldsymbol{x}_b^g$, and $\boldsymbol{x}_c^g$ which are different from $\boldsymbol{x}_i^g$ are randomly selected from the population to generate a mutant vector

$$\boldsymbol{v}_i^g = \boldsymbol{x}_a^g + F\left(\boldsymbol{x}_b^g - \boldsymbol{x}_c^g\right) \tag{1}$$

where $a \neq b \neq c \neq i$, $F \in (0, 1]$ is the scaling factor. Some other widely used mutation strategies are summarized in [31].

Subsequently, to produce a candidate offspring individual $\boldsymbol{u}_i^g$, the crossover between $\boldsymbol{x}_i^g$ and $\boldsymbol{v}_i^g$ is performed in the following way:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } \mathrm{rand}_j(0,1) \leq C_R \text{ or } j = j_{\mathrm{rand}} \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathrm{rand}_j(0, 1)$ is a uniformly distributed random number in the interval $[0, 1]$, j_{rand} is a random integer selected from $\{1, 2, \ldots, D\}$, and $\mathrm{CR} \in [0, 1]$ is the crossover rate.

Finally, the better individual is chosen from $\boldsymbol{x}_i^g$ and $\boldsymbol{u}_i^g$ according to the following formula to update the population:

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

where f is the objective function to be minimized.

B. Basic Underestimation Model

According to abstract convex theory [32], [33], a tight underestimation model of the objective function can always be built according to the supporting vectors of the given points (supporting points) [34]. Based on this conclusion, in DE, the underestimation model of the objective function can

be constructed by the supporting function of each individual in the current population. The supporting function of an individual $\boldsymbol{x}_i^g$ can be calculated by

$$h_i^g(\boldsymbol{x}) = \min_{j\in J}\left(f\left(\boldsymbol{x}_i^g\right) - C\left(x_{i,j}^g - x_j\right)\right) \tag{4}$$

where $J = \{1, 2, \ldots, D+1\}$, $x_{i,D+1}^g = 1 - \sum_{j=1}^{D} x_{i,j}^g$ is a slack variable to simplify the supporting function, and C is the slope control parameter of the supporting function [35].

From the supporting functions of the NP (population size) individuals, we can get NP estimation values $h_1^g(\boldsymbol{x}), h_2^g(\boldsymbol{x}), \ldots, h_{\mathrm{NP}}^g(\boldsymbol{x})$ for each point $\boldsymbol{x}$. As the supporting function is always below the objective function [34], the maximum estimation value which is the closest one to the actual function value is considered as the underestimation value of $\boldsymbol{x}$. Hence, the underestimation model of the objective function f can be described as

$$U(\boldsymbol{x}) = \max_{i\le \mathrm{NP}} h_i^g(\boldsymbol{x}). \tag{5}$$

Clearly, the underestimation model covers the whole search space since it is constructed based on all individuals in the current population. Therefore, the underestimation value of each point in the search space can be obtained by (5).

III. Proposed UMS

As reviewed in Section I, the most widely used multimutation DE variants choose the mutation strategy based on a probability model. However, building a probability model that ensures no strategy will be lost during the search process is not an easy task. The multistrategy coexistence technique simultaneously generates multiple candidate offsprings by employing several different mutation strategies. It is effective but computationally expensive as it needs to calculate the function value of multiple candidate offsprings. In addition, the surrogate model is usually employed to replace the time-consuming FEs because the computational cost taken to build the surrogate model is much cheaper compare with the real FE. Motivated by these findings, we propose the UMS, where a set of candidate offsprings are first produced for each target individual by using multiple mutation strategies. Then, the CAUM is built to obtain the underestimation value of each candidate offspring. The candidate offspring individuals are filtered according to the quality measured by the underestimation value, and only one is left as the offspring. Fig. 1 shows the flowchart of UMS.

A. Candidate Offspring Generation

Suppose that there are N mutation strategies in the mutation strategy pool. For the ith target individual $\boldsymbol{x}_i^g$ at the gth generation, N different mutant vectors $\boldsymbol{v}_i^{1,g}, \boldsymbol{v}_i^{2,g}, \ldots, \boldsymbol{v}_i^{N,g}$ are first created by utilizing each mutation strategy in the strategy pool. According to the crossover operation (2), N candidate offsprings $\boldsymbol{u}_i^{1,g}, \boldsymbol{u}_i^{2,g}, \ldots, \boldsymbol{u}_i^{N,g}$ are generated for the target individual $\boldsymbol{x}_i^g$.

B. Cheap Abstract Convex Underestimation Model

For each candidate offspring, we need to evaluate its quality. This is usually achieved by the surrogate model because the surrogate model is computationally cheaper than the FE, especially for expensive-to-evaluate problems [36]. Generally, the surrogate model can be classified as global surrogate models and local surrogate models [25]. In global-surrogate assisted metaheuristic algorithms, the surrogate model is built for the entire population [27], [37], while the surrogate model is just applied to evaluate some filtered or selected individuals in local-surrogate assisted metaheuristic algorithms [38], [39]. Similar to the majority of global-surrogate assisted methods, it will be computationally expensive if all individuals are applied to calculate supporting vectors to obtain the underestimation model of the whole search space. Therefore, a cheap underestimation model referred to as CAUM is presented in this section to reduce the computational cost. In CAUM, only some individuals are employed for the underestimation model construction. More details are described as follows.

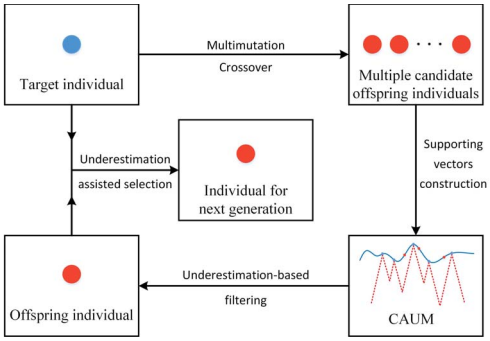


Fig. 1. Flowchart of the proposed UMS.

For the N candidate offsprings $\boldsymbol{u}_i^{1,g}, \boldsymbol{u}_i^{2,g}, \ldots, \boldsymbol{u}_i^{N,g}$ of the target individual $\boldsymbol{x}_i^g$, the center point $\boldsymbol{c}_i^g$ of these N candidate offsprings is first calculated by

$$c_{i,j}^g = \frac{\sum_{n=1}^{N} u_{i,j}^{n,g}}{N} \tag{6}$$

where $j = 1, 2, \ldots, D$, and $c_{i,j}^g$ denotes the jth variable of $\boldsymbol{c}_i^g$.

Then, the Euclidean distances between each candidate offspring and the center point $\boldsymbol{c}_i^g$ are calculated by

$$d_n^g = \sqrt{\sum_{j=1}^{D}\left(u_{i,j}^{n,g} - c_{i,j}^g\right)^2} \tag{7}$$

where d_n^g is the distance between the nth candidate offspring individual and the center point $\boldsymbol{c}_i^g$.

In order to select the individuals related to the candidate offsprings from the current population to construct the underestimation model, the Euclidean distances between the center point $\boldsymbol{c}_i^g$ and each individual in the population are also calculated based on (7). Let $d_{\max} = \max_{n=1,\ldots,N} d_n^g$, then all individuals with distances less than or equal to $d_{\max}$ are selected to construct the CAUM.

Suppose that there are M selected individuals $\boldsymbol{x}_1^g, \boldsymbol{x}_2^g, \ldots, \boldsymbol{x}_M^g$. The CAUM can be constructed based on the supporting vectors of the selected individuals, namely

$$U\left(\boldsymbol{u}_i^{n,g}\right) = C \max_{m\le M} \min_{j\in J}\left(V_{m,j}^g + u_{i,j}^{n,g}\right) \tag{8}$$

where

$$V_{m,j}^{g} = \frac{f(\boldsymbol{x}_m^g)}{C} - x_{m,j}^{g} \tag{9}$$

represents the jth variable of V_m^g; V_m^g is the support vector of $\boldsymbol{x}_m^g$; $x_{m,D+1}^g = 1 - \sum_{j=1}^{D} x_{m,j}^g$ is a slack variable used to simplify the supporting vector [34]; and C is the slope control factor of the supporting vectors [35]. Note that the value of $f(\boldsymbol{x}_m^g)$ will return to a large value when it equals to NaN or Inf. Also, if $M = 0$, that is, no individuals satisfy the condition that their distances to the center point should be less than or equal to $d_{\max}$, then $N(N < \mathrm{NP})$ individuals are randomly chosen from the current population to calculate the supporting vectors.

The underestimation value of each candidate offspring can be computed by (8). Moreover, the underestimation model (8) has been proven to always be below the objective function [33], [35], that is, the underestimation value of the candidate offspring is always lower than its actual function value, and can consequently be applied to measure the quality of the candidate offspring.

As described above, the underestimation value of each candidate offspring is directly calculated by the CAUM constructed by the supporting vectors of some selected individuals. The CAUM does not highly rely on the properties and characteristics of the objective function and thus can be applied to most of problems. The computational cost required by the CAUM is relatively small compared with that taken by the FEs of multiple candidate offsprings, particularly for the computationally expensive problems. Furthermore, the CAUM is removed after each iteration and reconstructed in the next iteration. The pseudocode of the CAUM is described in Algorithm 1 (see supplementary material).

C. Underestimation-Based Offspring Selection

According to the quality measured by the underestimation value instead of the objective function value, one offspring $\boldsymbol{u}_i^{g*}$ is chosen from the N candidate solutions by comparing the underestimation values $U(\boldsymbol{u}_i^{1,g})$, $U(\boldsymbol{u}_i^{2,g}), \ldots, U(\boldsymbol{u}_i^{N,g})$ calculated by (8). Similar to the greedy selection used in [20], the candidate solution with the lowest underestimation value is selected as the offspring individual.

In addition, the underestimation value of the offspring can also be used for population updating [40], [41]. Briefly, in the selection process of DE, we can judge whether the offspring will replace its corresponding target individual by comparing the underestimation value of the offspring with the function value of the target individual. Namely, the offspring can be directly discarded if its underestimation value is higher than the function value of the target individual, because the underestimation value of the offspring is always lower than its corresponding function value.

D. Overall Implementation

As the flowchart shown in Fig. 1, in the proposed UMS, a set of candidate offsprings are first generated by different mutation strategies for each target individual. Then some individuals in the current population are selected to build the CAUM. According to the underestimation value of each candidate offspring obtained by the CAUM, one offspring is chosen from the candidate offsprings. Finally, the population is updated by comparing the underestimation or function value of the offspring with the founction value of the target individual. The pseudocode of the proposed UMS is given in Algorithm 2 (see supplementary material). As shown in Algorithm 2, no mutation strategy will be lost since each mutation strategy generates a candidate offspring in the search process. In contrast, some mutation strategies may be lost in the probability model-based strategy self-adaptive technique if they obtain very low probabilities because of their poor performance in the previous stages, even though they may be more suitable for the subsequent stage. The computational cost of the proposed approach is smaller than that of the multistrategy coexistence method because the quality of each candidate offspring is measured by the underestimation value rather than the function value, and the underestimation information is also employed to guide the selection operation in the UMS. Like the existing local surrogate assisted techniques [38], [42], the CAUM is cheap since it is constructed just based on some relevant individuals rather than the whole population. In addition, the proposed UMS is generic. It can be incorporated into other EAs and used to implement hybrid algorithms by combining different EAs. An example of UMS is displayed in Fig. S-2 (see supplementary material).

The proposed UMS is inspired by the work in [20], and based on our previous work in [40] and [43], but it differs from them in the following major aspects.

1) In [20], the density probability is estimated by a CSM to evaluate the quality of each candidate offspring individual. In this paper, the candidate offspring individual is directly filtered according to the underestimation value of the objective function value that obtained by the CAUM. Moreover, the underestimation becomes tighter along with the population converges. Thus, the success rate of choosing the best offspring also increase since the underestimation is more accurate.
2) In addition to filter the candidate offsprings, the underestimation value is also employed to guide the population updating to reduce the computation cost in our proposed UMS; whereas in [20], the estimated density probability is just applied to select the best offspring individual from the candidate offspring individuals.
3) In [40], only a single mutation strategy is applied to generate an offspring. In [43], the offspring is created by the mutation strategy selected from the corresponding strategy pool according to the probability model. In contrast, multiple different mutation strategies are employed in the proposed UMS to simultaneously produce multiple candidate offspring individuals.
4) The underestimation model of UMS is different from those of [40] and [43]. In the UMS, the underestimation model is constructed based on the supporting vectors of all individuals related to the candidate offsprings, while the supporting vectors of some neighboring individuals of the offspring are calculated to built the underestimation model in [40] and [43].

5) The underestimation information is used to select the most promising offspring and guide the selection between the offspring and the target individual in the UMS. In [40], the underestimation is employed for the invalid region exclusion and local enhancement besides the selection guidance, while in [43], the underestimation is only used to measure the degree of convergence of the population for the evolutionary stage estimation.

E. Runtime Complexity Analysis

Suppose that N mutation strategies are applied in the strategy pool, the runtime complexity of the multiple candidate offspring generation is $O(\mathrm{NP} \cdot N \cdot D)$. For the CAUM construction, the runtime complexity is mainly determined by the distance calculation between each individual and the center point, which is $O(\mathrm{NP}^2 \cdot D)$ since Euclidean distance is employed. For the underestimation-based offspring selection, the runtime complexity is $O(\mathrm{NP} \cdot N)$. In summary, because the number of mutation strategy N is smaller than or equal to the population size NP, the total runtime complexity of the proposed UMS is $O(\mathrm{NP}^2 \cdot D \cdot G)$ over G generations.

From the above analysis, we can find that the runtime complexity of the proposed UMS depends on the problem dimensionality. Thus, it may not be applied in large scale problems (e.g., 1000-D). However, based on the studies in [20] and [44], the runtime complexity of UMS is relatively small when the FE is expensive. Therefore, the UMS can be accepted for the real-world problem with computationally expensive FEs. The analysis of the computational time for benchmark functions will be reported in Sections IV-B and IV-E. In addition, the effectiveness and efficiency of UMS for the real-world problem will be analyzed in Section IV-H.

IV. Experimental Study

A. Benchmark Functions and Settings

The CEC 2013 benchmark set is utilized to demonstrate the performance of the proposed UMS unless a change is mentioned. The benchmark set includes 28 functions, where F_1–F_5 are unimodal functions, F_6–F_{20} are multimodal functions, and F_{21}–F_{28} are composition functions. The other information on these functions is given in [45].

As Liang *et al.* [45] suggested, all algorithms are terminated when the number of FEs reaches $10\,000 \times D$. Each algorithm is independently performed 30 times for each benchmark problem, and the average and standard deviation of the function error value are calculated to measure the performance of each approach. Here, the function error is computed by $(f(\boldsymbol{x}) - f(\boldsymbol{x}^*))$, where $\boldsymbol{x}$ is the best solution achieved by the approach, and $\boldsymbol{x}^*$ represents the global optimal solution of the problem. The Wilcoxon signed-rank test is conducted at a 5% significance level to compare the significance between any two algorithms. The results are marked with "+," "−," and "≈" indicate that the UMS-based DE variant is significantly better than, worse than, and almost similar to the corresponding competitor, respectively. All algorithms were implemented in C++ language, and experiments were executed in a server with Windows Sever 2008 system, Intel Xeon 3.0 GHz CPU, and 48 GB of RAM.

TABLE I
Significance Test Results Between UMS-SHADE and Other SHADE Variants on 30-D CEC2013 Functions

UMS-SHADE vs	+	≈	−
SHADE1	17	7	4
SHADE2	14	6	8
SHADE3	17	5	6
SHADE4	17	6	5
SHADE-M	18	6	4

TABLE II
Significance Test Results Between UMS-SHADE and Other SHADE Variants on 50-D CEC2013 Functions

UMS-SHADE vs	+	≈	−
SHADE1	19	5	4
SHADE2	16	5	7
SHADE3	20	3	5
SHADE4	19	3	6
SHADE-M	21	4	3

B. Study on UMS-Based SHADE

In order to verify the performance of the proposed UMS, it is incorporated into SHADE and referred to as UMS-SHADE. In UMS-SHADE, four mutation strategies from JADE [14], that is, "DE/current-to-*p*best" with archive, "DE/rand-to-*p*best" with archive, DE/current-to-*p*best without archive, and DE/rand-to-*p*best without archive (see supplementary material), are employed for the candidate offspring generation. SHADE is chosen in this experiment because it is the best DE variant in the CEC 2013 competition.

For comparison study, the UMS-SHADE is compared with five SHADE variants, namely, SHADE1: DE/current-to-*p*best without archive, SHADE2: DE/current-to-*p*best with archive, SHADE3: DE/rand-to-*p*best without archive, SHADE4: DE/rand-to-*p*best with archive, and SHADE-M. In SHADE-M, the four mutation strategies used in UMS-SHADE are randomly chosen for the offspring generation. SHADE2 is the original version of SHADE presented in [46]. For all algorithms, the population size is set to 100. The parameter p in the four mutation strategies is randomly chosen from the range $[p_{\min}, 0.2]$, where $p_{\min}$ equals to $2/\mathrm{NP}$ [46]. Based on the study in [43], the slope control parameter C of supporting vectors is set to 10 000 for all benchmark functions.

The mean and standard deviation of the function errors achieved by the six SHADE algorithms for all 30-D and 50-D functions are summarized in Tables S-I and S-II (see supplementary material), respectively. For clarity, the overall best results are marked in bold font. The results in Tables S-I and S-II show that UMS-SHADE obtains the best result on the majority of functions. Moreover, the Wilcoxon signed-rank test results for 30-D and 50-D functions are, respectively, summarized in Tables I and II. As seen, for 30-D functions, UMS-SHADE, respectively, performs significantly better on 17, 14, 17, 17, and 18 out of 28 functions, and exhibit similar performance on 7, 6, 5, 6, and 6 functions, respectively, compared to SHADE1, SHADE2, SHADE3, SHADE4, and SHADE-M. SHADE1, SHADE2, SHADE3, SHADE4, and SHADE-M

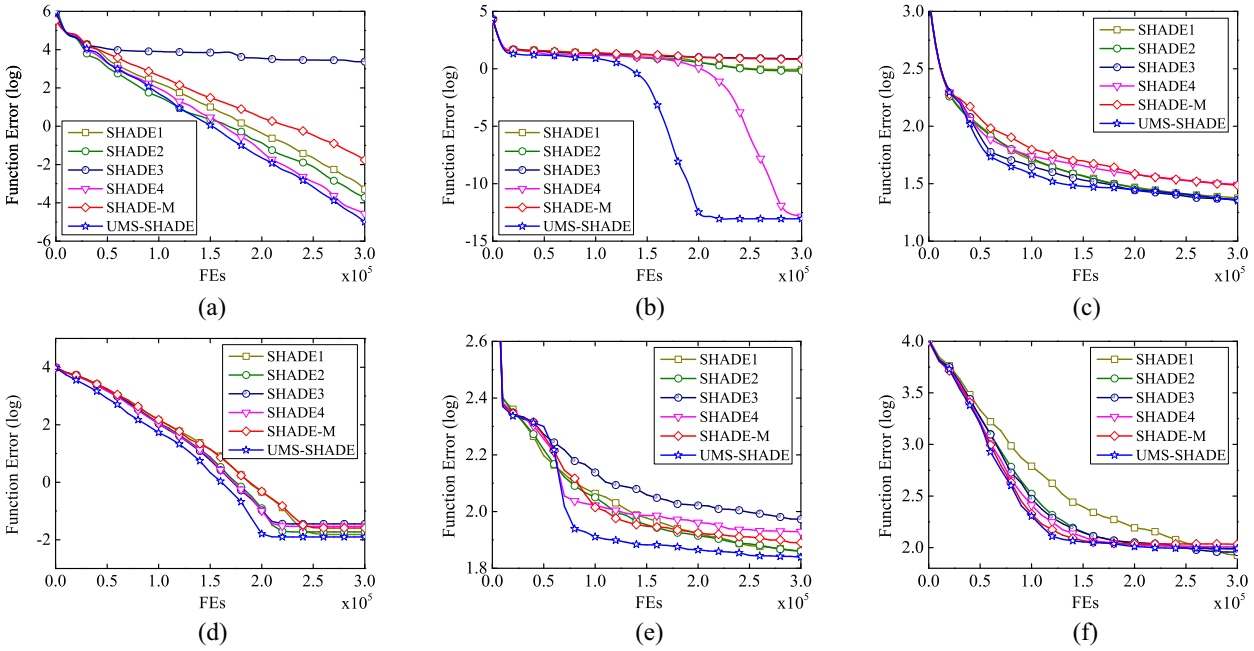


Fig. 2. Comparison of convergence speed between UMS-SHADE and other SHADE variants on six representative functions. (a) F_4. (b) F_6. (c) F_{12}. (d) F_{14}. (e) F_{18}. (f) F_{22}.

TABLE III
MULTIPLE-PROBLEM ANALYSIS BETWEEN UMS-SHADE AND OTHER SHADE VARIANTS ON 30-D CEC2013 FUNCTIONS

UMS-SHADE vs	R^+	R^-	p-value	Significant
SHADE1	200.0	53.0	0.0170	+
SHADE2	168.0	108.0	0.3614	≈
SHADE3	213.0	63.0	0.0225	+
SHADE4	200.0	53.0	0.0170	+
SHADE-M	190.5	40.5	0.0091	+

TABLE IV
MULTIPLE-PROBLEM ANALYSIS BETWEEN UMS-SHADE AND OTHER SHADE VARIANTS ON 50-D CEC2013 FUNCTIONS

UMS-SHADE vs	R^+	R^-	p-value	Significant
SHADE1	219.0	57.0	0.0137	+
SHADE2	196.0	80.0	0.0777	≈
SHADE3	273.5	51.5	0.0028	+
SHADE4	234.0	91.0	0.0544	≈
SHADE-M	281.0	19.0	0.0002	+

show remarkably better performance than UMS-SHADE on 4, 8, 6, 5, and 4 functions, respectively. For 50-D functions, UMS-SHADE is obviously superior to SHADE1, SHADE2, SHADE3, SHADE4, and SHADE-M on 19, 16, 20, 19, and 21 functions, respectively. UMS-SHADE also respectively exhibits similar performance on 5, 5, 3, 3, and 4 functions compared to SHADE1, SHADE2, SHADE3, SHADE4, and SHADE-M. SHADE1, SHADE2, SHADE3, SHADE4, and SHADE-M are remarkably superior to UMS-SHADE on 4, 7, 5, 6, and 3 functions, respectively.

Aside from the single-problem analysis based on the Wilcoxon signed-rank test, the multiple-problem analysis [47] is also performed based on the Wilcoxon signed-rank test at $\alpha = 0.05$ to compare UMS-SHADE with other algorithms over all benchmark functions. Tables III and IV report the results of multiple-problem analysis between UMS-SHADE and other SHADE variants on all 30-D and 50-D problems, respectively. In the table, R^+ and R^- are the sum of ranks when UMS-SHADE is better than and worse than the comparison algorithms, respectively. From the results presented in Table III, it is clear that UMS-SHADE achieves higher R^+ values than R^- values for 30-D functions compared to all competitors. According to the p-value, UMS-SHADE performs significantly better than SHADE1, SHADE3, SHADE4, and SHADE-M because their p-values are less than 0.05. UMS-SHADE obtains similar performance to SHADE2 as the p-value is larger than 0.05.

From Table IV, we can find that UMS-SHADE also provides better performance than all the other SHADE variants for 50-D functions as the R^+ values are remarkably higher than the R^- values. However, from the p-values, UMS-SHADE is significantly better than SHADE1, SHADE3, and SHADE-M. UMS-SHADE shows similar performance to SHADE2 and SHADE4 because their p-values are larger than 0.05.

Moreover, the convergence graphs of some representative functions (F_4, F_6, F_{12}, F_{14}, F_{18}, and F_{22}) at $D = 30$ are given in Fig. 2. Obviously, UMS-SHADE shows the fastest convergence speed for all functions except for F_{22}. For F_{22}, UMS-SHADE always converges faster than the competitors before 250 000 FEs, but it was surpassed by SHADE1 in the end. Moreover, from the curves of F_4, F_{12}, and F_{18}, we can see that the convergence speed of UMS-SHADE is initially slower than its competitors because the accuracy of the underestimation depends on the density of the selected individuals, which are used to construct the CAUM. In the beginning of the search, the population is relatively scattered, consequently leading to a high underestimation error. Since we use the underestimation value to measure the quality of each candidate offspring, the inaccurate underestimation value may influence the choice of offspring at the beginning. However, as the

TABLE V
SIGNIFICANCE TEST RESULTS BETWEEN UMS-BASED SHADE VARIANTS AND SHADE ON 30-D CEC2013 FUNCTIONS

	+	≈	−
UMS-SHADEo vs SHADE	14	7	7
UMS-SHADEm vs SHADE	14	5	9

TABLE VI
SIGNIFICANCE TEST RESULTS BETWEEN UMS-BASED SHADE VARIANTS AND SHADE ON 50-D CEC2013 FUNCTIONS

	+	≈	−
UMS-SHADEo vs SHADE	15	5	8
UMS-SHADEm vs SHADE	15	4	9

search proceeds, the population becomes more crowded and better underestimation value is obtained to select the more suitable offspring. This is the reason that UMS-SHADE gradually converges faster than the comparison algorithms.

The above results have demonstrated that UMS-SHADE performs better than the single mutation-based SHADE in the aspects of the solution's quality and convergence speed. However, the computational time is also important. To test this, we compare the runtime of SHADE2 (the original SHADE variant) and UMS-SHADE over some selected 30-D and 50-D CEC 2013 functions (F_1, F_4, F_5, and F_{11}). Following suggestions from [12] and [48], the runtime cost to reach the function error below a threshold value (0.0001) or reach $10\,000 \times D$ FEs is recorded for all functions. The ratio for the average runtime of UMS-SHADE divided by that of SHADE2 is calculated for comparison. The experimental results show that the ratio of 30-D and 50-D functions is 3.342 and 3.286, respectively. The comparison between UMS-SHADE and the methods that were also designed for computationally expensive problems will be reported in Section IV-E.

From the above analysis, it can be concluded that the proposed UMS significantly enhances the performance of the single mutation-based SHADE. The UMS is capable of obtaining the underestimate value of each candidate offspring effectively and promoting the cooperation of multiple mutation strategies for most of the benchmark functions. Accordingly, it can choose the most appropriate mutation strategy for different problem or in different evolutionary stages.

C. Contributions of UMS

In order to further verify the contributions of the proposed UMS, two UMS-SHADE variants (i.e., UMS-SHADEo and UMS-SHADEm) are designed to compare with the original SHADE [46]. In UMS-SHADEo, we just use one mutation strategy (i.e., DE/current-to-*p*best with archive) four times to generate four candidate offspings for each individual in each generation and choose the offspring by the UMS. UMS-SHADEm employs four very different mutations (i.e., DE/current-to-*p*best with archive, "DE/rand/1," "DE/best/1," and "DE/current-to-gr_best/1" [49]) to generate the candidate offsprings and determines the best offspring according to the UMS. All parameters of these three approaches are kept the same as in the original SHADE.

Table S-III (see supplementary material) reports the results of the comparison algorithms on all 30-D CEC 2013 problems. Obviously, the UMS-based SHADE methods are superior to SHADE on the majority of functions. Also, the significance test results obtained by the Wilcoxon signed-rank test are listed in Table V. As seen, UMS-SHADEo performs remarkably better than SHADE on 14 out of 28 functions and obtains results similar to SHADE on 7 functions. SHADE is obviously better than UMS-SHADEo on 7 functions. Although SHADE gets significantly better performance than UMS-SHADEm on 9 problems, UMS-SHADEm provides significantly better results than SHADE on 14 functions and gets equally good performance on 5 functions compared to SHADE.

The results of the 50-D CEC 2013 functions are presented in Table S-IV (see supplementary material). The data indicate that the UMS-based SHADE approaches also outperform SHADE on the majority of functions. In addition, the Wilcoxon signed-rank test results summarized in Table VI show that UMS-SHADEo provides obviously better performance than SHADE on 15 out of 28 problems and gets performance similar to SHADE on 5 problems. SHADE is significantly better than UMS-SHADEo on 8 functions. UMS-SHADEm achieves remarkably better performance on 15 functions and exhibits equally good performance on 4 functions compared to SHADE, while SHADE is significantly better than UMS-SHADEm on 9 problems.

The above results demonstrate that the superiority of the UMS-SHADE approach is due to the fact that the proposed UMS can choose the promising offspring from the multiple candidate offsprings, not due to the use of multiple diverse mutation strategies.

D. Comparison With CEC Winners

In this section, UMS-SHADE is compared with several CEC winners (i.e., L-SHADE [50], LSHADE-EpSin [51], LSHADE-ND [52], iLSHADE [53], and MC-SHADE [54]) in recent CEC competitions on CEC 2014 benchmark set. In the benchmark set, F_1–F_3 are unimodal functions, F_4–F_{16} are multimodal functions, F_{17}–F_{22} are hybrid functions, and F_{23}–F_{30} are composition functions. More information of these functions can be found in [55].

Tables S-V and S-VI (see supplementary material) summarize the mean and standard deviation results of the 30-D and 50-D CEC 2014 functions, respectively. It is obvious that UMS-SHADE outperforms the other five SHADE methods on most of the functions. In addition, the significance test results of 30-D and 50-D functions are listed in Tables VII and VIII, respectively. Clearly, compared to L-SHADE, LSHADE-EpSin, LSHADE-ND, iLSHADE, and MC-SHADE, UMS-SHADE achieves remarkably better performance on 16, 11, 16, 11, and 24 out of 30 functions for 30-D problems, respectively. UMS-SHADE also obtains the performance similar to L-SHADE, LSHADE-EpSin, LSHADE-ND, iLSHADE, and MC-SHADE on 12, 10, 12, 14, and 5 functions, respectively. L-SHADE, LSHADE-EpSin, LSHADE-ND, and iLSHADE significantly outperform UMS-SHADE on 2, 9, 2, and 5 functions, respectively. MC-SHADE is obviously superior

TABLE VII
SIGNIFICANCE TEST RESULTS BETWEEN UMS-SHADE AND ADVANCED SHADE VARIANTS ON 30-D CEC2014 FUNCTIONS

UMS-SHADE vs	+	≈	−
L-SHADE	16	12	2
LSHADE-EpSin	11	10	9
LSHADE-ND	16	12	2
iLSHADE	11	14	5
MC-SHADE	24	5	1

TABLE VIII
SIGNIFICANCE TEST RESULTS BETWEEN UMS-SHADE AND ADVANCED SHADE VARIANTS ON 50-D CEC2014 FUNCTIONS

UMS-SHADE vs	+	≈	−
L-SHADE	17	10	3
LSHADE-EpSin	13	6	11
LSHADE-ND	17	10	3
iLSHADE	15	8	7
MC-SHADE	23	5	2

TABLE IX
SIGNIFICANCE TEST RESULTS BETWEEN UMS-SHADE AND SURROGATE-ASSISTED OR UNDERESTIMATION-BASED ALGORITHMS ON 30-D CEC2013 FUNCTIONS

UMS-SHADE vs	+	≈	−
CSM-SHADE	14	6	8
ESMDE	25	1	2
GPEME	25	2	1
LLUDE	15	8	5
UMDE	14	7	7

to UMS-SHADE only on one function. For 50-D functions, UMS-SHADE significantly outperforms L-SHADE, LSHADE-EpSin, LSHADE-ND, iLSHADE, and MC-SHADE on 17, 13, 17, 15, and 23 functions, respectively. UMS-SHADE gets equally good performance on 10, 6, 10, 8, and 5 functions, respectively, compared to L-SHADE, LSHADE-EpSin, LSHADE-ND, iLSHADE, and MC-SHADE. However, L-SHADE, LSHADE-EpSin, LSHADE-ND, iLSHADE, and MC-SHADE are obviously better than UMS-SHADE on 3, 11, 3, 7, and 2 functions, respectively.

E. Comparison With Surrogate-Assisted and Underestimation-Based Algorithms

In this section, UMS-SHADE is compared with three advanced surrogate-assisted approaches (i.e., CSM-SHADE [20], ESMDE [28], and GPEME [27]) and two underestimation-based algorithms (i.e., LLUDE [40] and UMDE [43]) that were also designed for computationally expensive problems. In order to demonstrate the efficiency of UMS-SHADE, all algorithms are terminated when the runtime reaches 120 s. The parameter of all approaches are kept the same as in their original papers.

The mean and standard deviation of function errors achieved by these six algorithms are summarized in Table S-VII (see supplementary material). Clearly, UMS-SHADE performs better than the five approaches on the majority of problems. Moreover, the significance test results computed by the Wilcoxon signed-rank test are listed in Table IX. Specifically, UMS-SHADE respectively obtains obviously better performance than CSM-SHADE, ESMDE, GPEME, LLUDE, and UMDE on 14, 25, 25, 15, and 14 out of 28 problems. Also, UMS-SHADE exhibits the performance similar to CSM-SHADE, GPEME, LLUDE, and UMDE on 6, 2, 8, and 7 functions, respectively. ESMDE and UMS-SHADE provide the equally good performance only on one function. Compared to UMS-SHADE, CSM-SHADE, ESMDE, LLUDE, and UMDE is remarkably better on 8, 2, 5, and 7 functions, respectively. GPEME achieves obviously better performance than UMS-SHADE only on one function.

TABLE X
SIGNIFICANCE TEST RESULTS BETWEEN THE ADVANCED DE VARIANT AND ITS UMS-BASED VARIANT ON 30-D CEC2013 FUNCTIONS

	+	≈	−
UMS-CoDE vs CoDE	14	5	9
UMS-SaDE vs SaDE	15	5	8
UMS-JADE vs JADE	15	6	7

TABLE XI
SIGNIFICANCE TEST RESULTS BETWEEN THE ADVANCED DE VARIANT AND ITS UMS-BASED VARIANT ON 50-D CEC2013 FUNCTIONS

	+	≈	−
UMS-CoDE vs CoDE	15	3	10
UMS-SaDE vs SaDE	16	5	7
UMS-JADE vs JADE	14	7	7

F. Effect on Advanced DE Variants

In order to study the generality of the proposed UMS, the UMS is incorporated into three advanced DE approaches, i.e., CoDE [21], SaDE [17], and JADE [14] in this section. The UMS-based CoDE, SaDE, and JADE are denoted as UMS-CoDE, UMS-SaDE, and UMS-JADE, respectively. In UMS-CoDE and UMS-SaDE, the mutation strategies applied in their original algorithms are used for the candidate offsprings generation. Like UMS-SHADE, the four *p*best-based strategies [14] are also utilized in UMS-JADE. All parameters of the UMS-based methods are determined based on the corresponding original papers.

The results obtained by the above DE variants on 30-D CEC 2013 functions are reported in Table S-VIII (see supplementary material). From these data, we can see that the UMS-based DE algorithms obtain significantly better performance than their corresponding original DE algorithms on the majority of problems. Furthermore, the significance test results provided by the Wilcoxon signed-rank test are listed in Table X. Specifically, CoDE is obviously better than UMS-CoDE on 9 functions and exhibits similar performance to UMS-CoDE on 5 functions, while UMS-CoDE gets remarkably better performance than CoDE on 14 out of 28 functions. UMS-SaDE provides either better or similar performance on 20 functions. Concretely, the UMS significantly improves the performance of SaDE on 15 functions. SaDE is obviously superior to UMS-SaDE on 8 functions. For the other 5 functions, UMS-SaDE and SaDE obtain equally good performance. The proposed UMS improves JADE or performs uniformly well on 21 functions. As expected, UMS-JADE attains remarkably better results than JADE on 15 functions. JADE shows better performance than UMS-JADE on 7 functions. For the

TABLE XII
SIGNIFICANCE TEST RESULTS BETWEEN UMS-BASED DE VARIANTS AND ITS COMPETITORS ON 30-D CEC2013 FUNCTIONS

	+	≈	−
UMS-CoDE vs CSM-CoDE	14	6	8
UMS-JADE vs AP-JADE	16	4	8
UMS-JADE vs PM-JADE	14	6	8

other 6 functions, UMS-JADE and JADE obtain the same or similar performance.

Table S-IX (see supplementary material) further presents the results on 50-D CEC 2013 functions. Generally, it is difficult to solve for higher dimensional functions as the complexity of a function increases exponentially with its dimension. Nevertheless, the UMS-based DE methods significantly outperform their corresponding original DE variants on most of the functions. As the results listed in Table XI, the proposed UMS enhances either similar or remarkably better results on 18 out of 28 functions. Concretely, UMS-CoDE obtains significantly better performance than CoDE on 15 functions, while CoDE is significantly better than UMS-CoDE on 10 functions. UMS-SaDE provides significantly better or equal performance on 21 functions compared to SaDE. SaDE performs significantly better than UMS-SaDE on 7 functions. UMS-SaDE achieves significantly better results than SaDE on 16 functions. UMS-JADE either improves JADE or performs uniformly well on 21 functions. Specially, UMS-JADE performs significantly better on 14 functions and offers the equally good performance on 7 functions compared to JADE. JADE is significantly superior to UMS-JADE only on 7 functions.

G. Comparison With Other Multimutation Approaches

As mentioned in Section I, many multimutation approaches which are based on different techniques have been presented. In this section, the performance of UMS-CoDE is compared with the CSM-CoDE [20]. In CSM-CoDE, a CSM is built to obtain the density probability of each candidate offspring, which is used to filter the offspring. In addition, we also compare UMS-JADE with two multimutation-based methods, i.e., AP-JADE [56] and PM-JADE [56]. Both PM-JADE and AP-JADE are JADE variants with multiple mutation operators, and the mutation operator is chosen according to the probability matching (PM) in PM-JADE, while the adaptive pursuit (AP) in AP-JADE. All parameters of these comparison methods are kept the same as used in their original papers to ensure a fair comparison.

The results of the above algorithms on all 30-D CEC 2013 benchmark functions are given in Table S-X (see supplementary material). It is clear that the UMS-based DE algorithms are superior to the competitors on most of the problems. Moreover, the significance test results achieved by the Wilcoxon signed-rank test are reported in Table XII. Obviously, UMS-CoDE obtains either better or similar performance on 20 out of 28 functions compared to CSM-CoDE. More specifically, UMS-CoDE performs significantly better on 14 functions and obtains equally good performance on 6 functions, while CSM-CoDE is significantly superior to UMS-CoDE only on 8 functions. UMS-JADE also obtains the best results among these multimutation-based JADE variants on the majority of functions. To be specific, UMS-JADE provides remarkably better results than AP-JADE and PM-JADE on 16 and 14 out of 28 functions, respectively. In addition, compared to AP-JADE and PM-JADE, UMS-JADE attains equally good performance on 4 and 6 functions, respectively. AP-JADE and PM-JADE is obviously better than UMS-JADE on 8 and 8 functions, respectively.

H. Comparison in Real-World Application Problem

In the previous sections, the performance of the proposed UMS has been verified via benchmark functions. In this section, the UMS-CoDE is applied to a real-world problem, namely, the PSP problem [57], [58], which is important in the computational biology.

The PSP problem aims to find the native 3-D structure of a protein according to its amino acid sequence. In the light of the thermodynamics hypothesis [59], the structure associated with the global minimum of an energy function is believed to be the most native-like, that is, the native structure of a protein obtains the lowest free energy conformation. Therefore, the PSP problem can be defined as an optimization problem of the energy function. However, it is not easy to find the global minimum of the energy function as it includes numerous local minima. To address this problem, a variety of algorithms have been presented over the last few decades [60]–[62]. Some detailed surveys on the PSP methods can be found in [63] and [64].

1) Effectiveness of UMS: The UMS-CoDE is compared with DE-M and CoDE [21] to verify the effectiveness of UMS for the real-world problem. In DE-M, three mutation strategies (i.e., DE/rand/1, "DE/rand/2," and "DE/current-to-rand/1") used in CoDE are applied and a strategy is randomly chosen for each target individual to create an offspring. The performance of these three algorithms is evaluated by six structurally diverse protein sequences of varying lengths. Moreover, in order to obtain a high prediction accuracy, the fragment assembly [65] with the length of 9 residues is utilized in each algorithm. The fragment library is downloaded from ROBETTA full-chain PSP server (http://robetta.bakerlab.org). For each query sequence, the homologous fragments are excluded from the template library. All algorithms are stopped when the number of energy FEs reaches 300 000, and each protein is predicted over 30 independent runs. The energy function, namely, Rosetta score3 [66], which contains all possible energy terms except for hydrogen bonding, is used in this experiment. Detailed description of Rosetta score3 can be found in [66]. To measure the quality of the predicted structure, the root-mean-squared-deviation (RMSD), which is the Euclidean distance between C_α atoms in the predicted structure and the native structure after optimal rigid-body superimposition, is applied in the following experiments. The parameter settings used in these three algorithms are the same as in CoDE. However, different from the benchmark functions, the slope control factor C of the UMS is trained in the initial generation for the real-world problem. More details can be found in the supplementary material.

TABLE XIII
MINIMUM RMSD OBTAINED BY DE-M, CODE, AND UMS-CODE

PDB ID	DE-M	CoDE	UMS-CoDE
1ENH	2.77	2.27	**1.50**
1BBO	6.70	5.82	**4.69**
1AIL	5.22	5.71	**3.75**
1CC5	6.75	6.43	**5.22**
1GB1	6.37	5.90	**3.12**
1I6C	5.82	4.76	**4.43**

TABLE XIV
MEAN AND STANDARD DEVIATION VALUE OF RMSD OBTAINED BY DE-M, CODE, AND UMS-CODE

PDB ID	DE-M	CoDE	UMS-CoDE
1ENH	3.18(0.26)$^+$	2.47(0.25)$^+$	**1.66(0.15)**
1BBO	7.12(0.38)$^+$	5.93(0.21)$^+$	**5.30(0.34)**
1AIL	5.92(0.59)$^+$	5.86(0.17)$^+$	**3.97(0.18)**
1CC5	7.58(0.91)$^+$	7.27(0.53)$^+$	**5.60(0.25)**
1GB1	6.74(0.34)$^+$	6.50(0.52)$^+$	**3.41(0.15)**
1I6C	5.76(0.43)$^+$	5.21(0.42)$^+$	**4.78(0.21)**

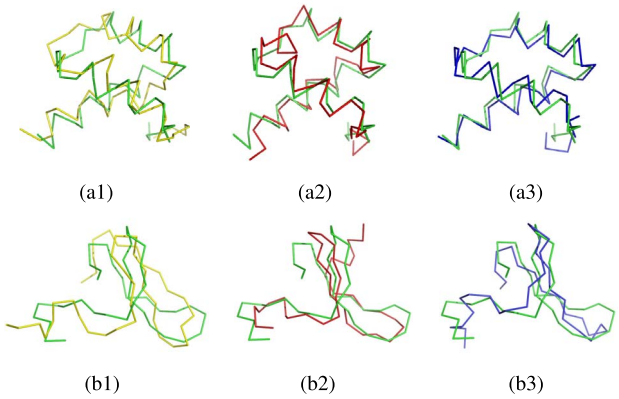


Fig. 3. Comparison of the structure predicted by DE-M (yellow), CoDE (red), and UMS-CoDE (blue) with the native structure (green) on two proteins. (a1) 1ENH(DE-M). (a2) 1ENH(CoDE). (a3) 1ENH(UMS). (b1) 1I6C(DE-M). (b2) 1I6C(CoDE). (b3) 1I6C(UMS).

The minimum RMSD of each protein achieved by DE-M, CoDE, and UMS-CoDE are given in Table XIII. From the data, we can find that UMS-CoDE provides the best results for all proteins. In addition, a comparison between the predicted structures with the minimum RMSD and the native structures on two representative proteins is displayed in Fig. 3, where the structures predicted by UMS-CoDE are marked with UMS. The degree of overlap between the predicted structure and the native structure indicates the prediction accuracy. Clearly, UMS-CoDE provides more accurate structures than its competitors for the 2 proteins. The predicted structure of the other proteins can be found in Fig. S-3 (see supplementary material).

Table XIV lists the mean and standard deviation results of RMSD achieved by the three algorithms. Obviously, UMS-CoDE achieves better results than DE-M and CoDE on all proteins. The results obtained by the Wilcoxon signed-rank test also indicate that UMS-CoDE performs significantly better than DE-M and CoDE on all proteins.

In addition, Fig. S-4 (see supplementary material) depicts the RMSD of each conformation change along with the energy on three representative proteins. Clearly, UMS-CoDE can always capture the conformations with lower energy and RMSD compared to CoDE. The figure also indicates that the sampling capability of UMS-CoDE (more points near the lower left corner of the figure) is better than that of CoDE as UMS-CoDE can guide substantial conformation to converge to more promising regions. All these improvements can be attributed to the proposed UMS which can choose more suitable conformation generation strategies in different search stages.

TABLE XV
MINIMUM RMSD OBTAINED BY CSM-CODE, ESMDE, GPEME, LLUDE, UMDE, AND UMS-CODE

PDB ID	CSM-CoDE	ESMDE	GPEME	LLUDE	UMDE	UMS-CoDE
1ENH	1.63	2.89	2.75	1.64	1.59	**1.50**
1BBO	4.81	6.30	6.89	4.86	4.53	**4.31**
1AIL	**3.58**	6.08	5.97	4.09	3.75	3.60
1CC5	5.28	7.33	7.38	5.62	5.25	**5.19**
1GB1	3.19	5.17	6.21	5.77	2.89	**2.72**
1I6C	4.62	7.74	7.69	**4.06**	4.59	4.38

TABLE XVI
MEAN AND STANDARD DEVIATION VALUE OF RMSD OBTAINED BY CSM-CODE, ESMDE, GPEME, LLUDE, UMDE, AND UMS-CODE

PDB ID	CSM-CoDE	ESMDE	GPEME	LLUDE	UMDE	UMS-CoDE
1ENH	1.73(0.09)$^+$	3.15(0.46)$^+$	3.16(0.30)$^+$	2.42(0.37)$^+$	1.67(0.13)$^+$	**1.63(0.13)**
1BBO	5.51(0.84)$^+$	7.35(0.88)$^+$	7.20(0.41)$^+$	5.54(0.68)$^+$	5.40(0.67)$^+$	**5.19(0.66)**
1AIL	3.84(0.19)$^\approx$	7.04(0.69)$^+$	6.50(0.45)$^+$	4.74(0.38)$^+$	3.89(0.17)$^+$	**3.82(0.16)**
1CC5	5.69(0.27)$^+$	8.25(0.51)$^+$	8.04(0.39)$^+$	5.84(0.29)$^+$	**5.58(0.13)**$^-$	5.61(0.27)
1GB1	3.74(0.36)$^+$	6.26(0.67)$^+$	6.72(0.44)$^+$	6.18(0.44)$^+$	3.52(0.49)$^+$	**3.38(0.49)**
1I6C	4.84(0.28)$^+$	8.33(0.41)$^+$	8.11(0.29)$^+$	4.78(0.30)$^\approx$	4.81(0.26)$^+$	**4.77(0.23)**

2) Efficiency of UMS: In order to demonstrate the efficiency of the proposed UMS for the real-word problem, we further compare UMS-CoDE with CSM-CoDE [20], ESMDE [28], GPEME [27], LLUDE [40], and UMDE [43]. The parameters of UMS-CoDE are kept the same as in Section IV-H1. For the other five comparison methods, their parameters are set based on the corresponding original papers, while the slope control factor C in LLUDE and UMDE is calculated based on the initial generation (see supplementary material). For each approach and each protein, 30 independent predictions are conducted with 2 h as the stop rule according to the runtime of Rosetta [66].

Table XV summarizes the minimum RMSD of each algorithm. From the results, we can see that UMS-CoDE is superior to the competitors on 4 out of 6 proteins. CSM-CoDE and LLUDE obtains better results than the competitors only on one protein. ESMDE, GPEME, and UMDE cannot outperform the competitors on any proteins. In addition, the mean and standard deviation values of RMSD given in Table XVI also indicate UMS-CoDE performs better on most of the proteins. In the light of the Wilcoxon signed-rank test results, UMS-CoDE achieves significantly better performance than CSM-CoDE and LLUDE on five proteins. Compared to ESMDE and GPEME, UMS-CoDE obtains obviously better performance on all proteins. UMDE shows remarkably better performance than UMS-CoDE on one protein, but UMS-CoDE exhibits significantly better performance than UMDE on five proteins.

V. CONCLUSION

The performance of DE crucially depends on the appropriate mutation strategies. Although various mutation strategies

have been presented for DE, it is not trivial to choose the best one among them for a specific problem or different running stages because different strategies have different features. In this paper, we proposed a multimutation strategy based on the CAUM. In this approach, the quality of the candidate offsprings generated by multiple mutation strategies are evaluated by the CAUM, which is built according to the individuals related to the candidate offsprings. Among the candidate offsprings, only one is chosen as the offspring. In this way, the proposed approach can ensure that every mutation strategy has a chance of being applied during the search process. The proposed approach can also promote the cooperation of multiple mutation strategies to generate better offsprings in the different stages. Particularly, the proposed approach can be incorporated into other DE variants or other EAs to implement a method with multiple mutation strategies.

The proposed approach is incorporated into several advanced single and multiple mutation-based DE variants (i.e., SHADE, CoDE, SaDE, and JADE) and denoted as UMS-based algorithms. Their performance is evaluated by comparing with their corresponding original DE variants and some advanced DE variants over the CEC 2013 and 2014 benchmark sets. Experimental results indicate that the UMS-based algorithms achieve remarkably better performance than the competitors on the majority of problems. In addition, the effectiveness and efficiency of the UMS-based algorithm for the real-world problem is tested over the PSP problem. The results show that the UMS-based algorithm can obtain more reasonable structures with lower RMSD compared to its competitors.

In this paper, the CAUM is used to calculate the underestimation to measure the quality of each candidate offspring. However, more significant information may be obtained from the CAUM, such as the crowding degree of the population and the status of some individuals. In the future work, we intend to extract more information from the CAUM to guide the search process of DE. Moreover, the runtime complexity of DE is highly related to the population size. The population reduction technique such as [50], [67], and [68] will be an important method to reduce the computational cost of UMS-based DE algorithms if a fixed number of individuals are applied to construct the CAUM. Also, combining several superior DE methods or other advanced EAs through the proposed UMS is another work worth future study.

ACKNOWLEDGMENT

The authors would like to thank Dr. W. Gong and Dr. R. Mallipeddi for providing the source code of CSM-CoDE and ESMDE, respectively. They are also grateful to Dr. P. N. Suganthan and Dr. B. Liu for their valuable suggestions on the experiments and the implementation of GPEME, respectively.

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] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Comput.*, vol. 14, no. 11, pp. 1187–1207, Sep. 2010.

[3] G.-J. 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. Bioinformat.*, vol. 14, no. 6, pp. 1288–1301, Nov./Dec. 2017.

[4] B. Bošković and J. Brest, "Differential evolution for protein folding optimization based on a three-dimensional AB off-lattice model," *J. Mol. Model.*, vol. 22, no. 10, p. 252, Oct. 2016.

[5] M. Oliveira, B. Borguesan, and M. Dorn, "SADE-SPL: A self-adapting differential evolution algorithm with a loop structure pattern library for the PSP problem," in *Proc. IEEE Congr. Evol. Comput.*, San Sebastian, Spain, 2017, pp. 1095–1102.

[6] Y.-L. Li *et al.*, "Fast micro-differential evolution for topological active net optimization," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1411–1423, Jun. 2016.

[7] 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.

[8] V. Santucci, M. Baioletti, and A. Milani, "Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 682–694, Oct. 2016.

[9] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 61–106, 2010.

[10] 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.

[11] 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.

[12] 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.

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

[14] J. Q. 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.

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

[16] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 716–727, Apr. 2015.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[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] Y. Wang, Z.-Z. Liu, J. Li, H.-X. Li, and G. G. Yen, "Utilizing cumulative population distribution information in differential evolution," *Appl. Soft Comput.*, vol. 48, pp. 329–346, Nov. 2016.

[23] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 25–32.

[24] A. Sóbester, A. I. J. Forrester, D. J. J. Toal, E. Tresidder, and S. Tucker, "Engineering design applications of surrogate-assisted optimization techniques," *Optim. Eng.*, vol. 15, no. 1, pp. 243–265, 2014.

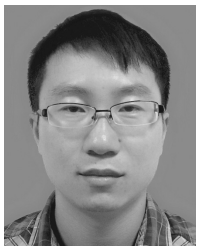
[25] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 644–660, Aug. 2017.

[26] L. Zhou, A. Zhou, G. Zhang, and C. Shi, "An estimation of distribution algorithm based on nonparametric density estimation," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 1597–1604.

[27] B. Liu, Q. Zhang, and G. G. E. Gielen, "A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, Apr. 2014.

[28] R. Mallipeddi and M. Lee, "An evolving surrogate model-based differential evolution algorithm," *Appl. Soft Comput.*, vol. 34, pp. 770–787, Sep. 2015.

[29] J. Müller and R. Piché, "Mixture surrogate models based on Dempster–Shafer theory for global optimization problems," *J. Glob. Optim.*, vol. 51, no. 1, pp. 79–104, Sep. 2011.
[30] I. Poikolainen, F. Neri, and F. Caraffini, "Cluster-based population initialization for differential evolution frameworks," *Inf. Sci.*, vol. 297, pp. 216–235, Mar. 2015.
[31] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Inf. Sci.*, vol. 181, no. 12, pp. 2488–2511, Jun. 2011.
[32] A. M. Rubinov, *Abstract Convexity and Global Optimization, of Nonconvex Optimization and Its Applications*. Dordrecht, The Netherlands: Kluwer Academic, 2000.
[33] G. Beliakov, "Geometry and combinatorics of the cutting angle method," *Optimization*, vol. 52, nos. 4–5, pp. 379–394, Aug. 2003.
[34] G. Beliakov and K. F. Lim, "Challenges of continuous global optimization in molecular structure prediction," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1198–1213, Sep. 2007.
[35] G. Beliakov, "Extended cutting angle method of global optimization," *Pac. J. Optim.*, vol. 4, no. 1, pp. 152–176, Jan. 2008.
[36] F. Neri, X. del Toro Garcia, G. L. Cascella, and N. Salvatore, "Surrogate assisted local search in PMSM drive design," *COMPEL Int. J. Comput. Math. Elect. Electron. Eng.*, vol. 27, no. 3, pp. 573–592, 2008.
[37] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.
[38] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 1, pp. 66–76, Jan. 2007.
[39] S. Z. Martinez and C. A. C. Coello, "Combining surrogate models and local search for dealing with expensive multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Caucún, Mexico, 2013, pp. 2572–2579.
[40] X.-G. Zhou, G.-J. Zhang, X.-H. Hao, D.-W. Xu, and L. Yu, "Enhanced differential evolution using local Lipschitz underestimate strategy for computationally expensive optimization problems," *Appl. Soft Comput.*, vol. 48, pp. 169–181, Nov. 2016.
[41] 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, pp. 132–149, Nov. 2016.
[42] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, Apr. 2003.
[43] X.-G. Zhou and G.-J. Zhang, "Abstract convex underestimation assisted multistage differential evolution," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2730–2741, Sep. 2017.
[44] 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.
[45] 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., Dept. Econ., Nanyang Technol. Univ., Singapore, Rep. 201212, 2013.
[46] 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.
[47] 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.
[48] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 89–99, Feb. 2013.
[49] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.
[50] 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.
[51] 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.
[52] 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.
[53] 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.
[54] 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.
[55] 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, Rep. 201311, Dec. 2013.
[56] 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, p. 5364–5386, Dec. 2011.
[57] Y. Zhang, "Progress and challenges in protein structure prediction," *Current Opinion Struct. Biol.*, vol. 18, no. 3, pp. 342–348, Jun. 2008.
[58] K. A. Dill and J. L. MacCallum, "The protein-folding problem, 50 years on," *Science*, vol. 338, no. 6110, pp. 1042–1046, 2012.
[59] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, no. 4096, pp. 223–230, 1973.
[60] S. M. Venske, R. A. Goncalves, E. M. Benelli, and M. R. Delgado, "ADEMO/D: An adaptive differential evolution for protein structure prediction problem," *Expert Syst. Appl.*, vol. 56, pp. 209–226, Sep. 2016.
[61] M. A. Rashid, F. Khatib, M. T. Hoque, and A. Sattar, "An enhanced genetic algorithm for ab initio protein structure prediction," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 627–644, Aug. 2016.
[62] X.-H. Hao, G.-J. Zhang, X.-G. Zhou, and X.-F. Yu, "A novel method using abstract convex underestimation in ab-initio protein structure prediction for guiding search in conformational feature space," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 13, no. 5, pp. 887–900, Sep. 2016.
[63] Y. Zhang, "Protein structure prediction: When is it useful?" *Current Opinion Struct. Biol.*, vol. 19, no. 2, pp. 145–155, Apr. 2009.
[64] J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano, "Critical assessment of methods of protein structure prediction: Progress and new directions in round XI," *Proteins*, vol. 84, no. 1, pp. 4–14, Sep. 2016.
[65] D. Xu and Y. Zhang, "Toward optimal fragment generations for ab initio protein structure assembly," *Proteins*, vol. 81, no. 2, pp. 229–239, Feb. 2013.
[66] C. A. Rohl, C. E. Strauss, K. M. Misura, and D. Baker, "Protein structure prediction using Rosetta," *Method Enzymol.*, vol. 383, pp. 66–93, Dec. 2004.
[67] M. Z. Ali, N. H. Awad, P. N. Suganthan, and R. G. Reynolds, "An adaptive multipopulation differential evolution with dynamic population reduction," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2768–2779, Sep. 2017.
[68] J. Brest and M. S. Maučec, "Population reduction differential evolution with multiple mutation strategies in real world industry challenges," *Swarm Evol. Comput.*, vol. 29, no. 3, pp. 228–247, Sep. 2008.



Xiao-Gen Zhou is currently pursuing the Ph.D. degree in control science and engineering 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.

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.