Differential Evolution With Two-Level Parameter Adaptation

Wei-Jie Yu, *Student Member, IEEE,* Meie Shen, Wei-Neng Chen, *Member, IEEE,* Zhi-Hui Zhan, *Member, IEEE,* Yue-Jiao Gong, *Student Member, IEEE,* Ying Lin, *Member, IEEE,* Ou Liu, and Jun Zhang, *Senior Member, IEEE*

***Abstract*—The performance of differential evolution (DE) largely depends on its mutation strategy and control parameters. In this paper, we propose an adaptive DE (ADE) algorithm with a new mutation strategy DE/lbest/1 and a two-level adaptive parameter control scheme. The DE/lbest/1 strategy is a variant of the greedy DE/best/1 strategy. However, the population is mutated under the guide of multiple locally best individuals in DE/lbest/1 instead of one globally best individual in DE/best/1. This strategy is beneficial to the balance between fast convergence and population diversity. The two-level adaptive parameter control scheme is implemented mainly in two steps. In the first step, the population-level parameters F_p and CR_p for the whole population are adaptively controlled according to the optimization states, namely, the exploration state and the exploitation state in each generation. These optimization states are estimated by measuring the population distribution. Then, the individual-level parameters F_i and CR_i for each individual are generated by adjusting the population-level parameters. The adjustment is based on considering the individual's fitness value and its distance from the globally best individual. This way, the parameters can be adapted to not only the overall state of the population but also the characteristics of different individuals. The performance of the proposed ADE is evaluated on a suite of benchmark functions. Experimental results show that ADE generally outperforms four state-of-the-art DE variants on different kinds of optimization problems. The effects of ADE components, parameter properties of ADE, search behavior of ADE, and parameter sensitivity of ADE are also studied. Finally, we investigate the capability of ADE for solving three real-world optimization problems.**

***Index Terms*—Adaptive parameter control, differential evolution (DE), global optimization.**

Manuscript received October 6, 2012; revised February 22, 2013, June 7, 2013; accepted August 6, 2013. Date of publication September 5, 2013; date of current version June 12, 2014. This work was supported in part by the National High-Technology Research and Development Program (863 Program) of China No. 2013AA01A212, in part by the NSFC for Distinguished Young Scholars 61125205, in part by the NSFC No. 61332002, No. 61300044, and No. 61070004. This paper was recommended by Associate Editor K.-C. Tan.

W.-J. Yu, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, and J. Zhang are with Sun Yat-Sen University, Guangzhou 510275, China, with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China, with the Engineering Research Center of Supercomputing Engineering Software, Ministry of Education, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, China (e-mail: junzhang@ieee.org).

M. Shen is with the School of Computer Science, Beijing Information Science and Technology University, Beijing, China.

Y. Lin is with the Department of Psychology, Sun Yat-Sen University, Guangzhou 510275, China.

O. Liu is with Hong Kong Polytechnic University, Hung Hom, Hong Kong.

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.2013.2279211

I. Introduction

DIFFERENTIAL evolution (DE), first proposed by Storn and Price [1], [2], is a simple and efficient evolutionary algorithm for global optimization. It has been successfully applied to a variety of numerical optimization problems [3]–[9] and real-world applications such as signal processing [10], robotic system control [11], and wireless sensor networks [12]. Theoretical studies on DE such as convergence analysis [13] and population-dynamics analysis [14] have also been conducted by some recent work.

A classic DE algorithm involves three general evolutionary operators, i.e., mutation, crossover, and selection, which are associated with certain control parameters. The values of the parameters greatly influence the convergence speed and population diversity. Therefore, how to choose an appropriate parameter setting to improve the performance of the algorithm has become a significant and promising research topic in DE.

A large amount of research work has been conducted to analyze the effects of these control parameters and suggest suitable parameter settings [2], [15], [16]. However, optimal parameter settings *ad hoc* to a specific problem are often based on *a priori* or empirical knowledge, and there exists no single value being good for all types of problems. For example, a small crossover probability CR is suitable for separable functions while a large one is effective for nonseparable functions [17]. Thus, many studies have been undertaken on parameter control for improved DE, where parameters are automatically adjusted at runtime.

In the literature, the works on parameter control can be mainly classified into two kinds of strategies. The first one is population-level parameter control and the second one is individual-level parameter control.

In the earlier works of parameter control for DE, the control parameters are usually applied at a population level. All individuals in the population are associated with the same parameter values. Two kinds of information can be used to adjust the parameter values for the whole population. The first kind is some deterministic factors such as generation number or fitness evaluations number [18]. Parameter control based on such factors is easy to implement, but may not be effective enough since it does not utilize any search feedback information from the evolution. The second kind is some form of feedback derived from the search process that can be in

one of different evolutionary states. Examples of this feedback information are the relative fitness values over individuals [19], [20] and population diversity [21], [22].

In recent years, more researchers apply the adaptive control parameters at an individual level [23]–[30]. Each individual maintains its own set of parameters that are adapted during the optimization process. Most of these adaptive approaches are based on the idea that the parameters that can lead to good individuals should be propagated throughout the population. There are various methods for adaptively updating the parameter values for each individual among these approaches. The related works will be reviewed in Section II-B.

It can be seen that most existing methods for parameter adaptation only take the population-level strategy or individual-level strategy into consideration. By themselves, however, the population-level and the individual-level parameter adaptations can make use of only one aspect of the evolutionary process information, that is, the state of the whole population or the characteristics of single individual. For the use of only population-level strategy, some deterministic factors or feedback information from the population are utilized to control parameter values. Nevertheless, since all the individuals share the same population-level parameters without diversity, the diversity of population search behavior may also be insufficient. For the use of only individual-level strategy, each individual is associated with its own parameters that are controlled based on individual characteristics instead of the overall population information. Therefore, this kind of strategy cannot quickly adjust the parameters to adapt to different optimization states.

How to utilize both of the two kinds of evolutionary process information to design a more efficient parameter adaptation strategy is still a challenging and significant task. So far as we know, in the literature, there is no reported work on combining both population-level and individual-level parameter control. In this paper, we propose a novel two-level adaptive parameter control scheme, which is an extension of our previous work [31], utilizing both population and individual information to provide a more comprehensive description of the evolutionary process. Thus, the proposed scheme can tackle the limitations of using only population-level or individual-level strategy. In this scheme, the parameters of the algorithm are adaptively controlled at two levels. At the population level, the population-level parameters shared by the whole population are first controlled. In each generation, the population-level parameters are updated according to different optimization states that are estimated by measuring the population distribution. In order to fit the characteristics of different individuals, the individual-level parameter control is then performed. The individual-level parameters assigned to each individual are generated by further adjusting the population-level parameters. The adjustment is based on considering the individual's fitness value and its distance from the best individual. Consequently, the two-level parameter adaptation scheme can provide adaptive parameters more effectively. Through population-level control, the proposed scheme is able to quickly adjust parameters to match the overall population state. Moreover, the resulting parameters can be further fine-turned to adapt to the characteristics of different individuals through individual-level control.

This two-level adaptive parameter control scheme is applied to DE with a new mutation strategy called DE/lbest/1, which is a variant of the classic DE/best/1. In traditional DE, DE/best/1 utilizes the globally best solution to guide the mutation, and can thus converge very fast on simple unimodal problems. However, such a greedy strategy may suffer from premature convergence when it deals with complex multimodal problems. By contrast, in the DE/lbest/1 mutation strategy, the population is attracted by multiple locally best solutions instead of a single globally best solution. Therefore, our proposed mutation strategy is helpful for balancing the exploration and exploitation abilities of the algorithm. Although there exist DE mutation strategies such as DE/current-to-*p*best/1 in JADE [26] utilizing multiple good solutions, our DE/lbest/1 differs from those strategies in the following major aspects.

1) The population of DE/lbest/1 is divided into a predefined number of nonoverlapping groups, whose number and members are kept unchanged at runtime.
2) In DE/lbest/1, multiple locally best solutions are selected from different groups of solutions, each of which is mutated under the guide of its respective locally best solution.

Since DE/lbest/1 utilizes the locally best solution which only directly attracts the individuals in the same group with it, the entire population is in fact attracted by multiple good solutions chosen from different groups. Therefore, it is less likely for DE/lbest/1 that the entire population is attracted to a specific region in the search space, and the problem of premature convergence can be better alleviated for DE/lbest/1.

Combining the DE/lbest/1 strategy with two-level adaptive parameter control, we develop a novel adaptive DE algorithm (ADE), the robustness and efficiency of which are further enhanced. The proposed ADE is tested on different types of benchmark functions and three real-world optimization problems. The performance of ADE compares favorably with four state-of-the-art DE variants.

The remainder of this paper is organized as follows. Section II reviews the DE algorithm and the related works on parameter control methods and variants of mutation strategies for DE. Section III describes the proposed ADE algorithm in detail, including the DE/lbest/1 mutation strategy, the two-level parameter control scheme, and the runtime complexity of ADE. In Section IV, the ADE algorithm is compared with four state-of-the-art DE variants on a suite of benchmark functions and three real-world optimization problems. The experimental results are also discussed. In addition, the effects of ADE components, parameter properties of ADE, search behavior of ADE, and parameter sensitivity of ADE are studied in this section. Finally, Section V draws the conclusions.

II. DE Algorithm and Related Works

A. Differential Evolution (DE) Algorithm

DE is a population-based stochastic algorithm designed for global numerical optimization. Similar to other EAs, DE

searches for a global optimum in the search space with a population of vectors $\{\mathbf{x}_i^g = [x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g], i = 1, 2, ..., NP\}$, where g denotes the current generation, D is the dimension of the search space, and NP is the population size. In generation $g = 0$, the jth component of the ith vector can be initialized as

$$x_{i,j}^0 = x_{\min,j} + \text{rand}(0, 1) \cdot (x_{\max,j} - x_{\min,j}) \tag{1}$$

where rand(0,1) is a uniform random number on the interval [0,1], and $x_{\min,j}$, $x_{\max,j}$ are the prescribed minimum and maximum bounds of the jth dimension, respectively. After initialization, DE enters an evolutionary process which includes mutation, crossover, and selection operations.

1) *Mutation:* In each generation g, the mutation operation is applied to each individual $\mathbf{x}_i^g$ (also called the target vector) to create its corresponding mutant vector $\mathbf{v}_i^g$. The five frequently used mutation strategies are listed as follows:

DE/rand/1

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g). \tag{2}$$

DE/current-to-best/1

$$\mathbf{v}_i^g = \mathbf{x}_i^g + F \cdot (\mathbf{x}_{\text{best}}^g - \mathbf{x}_i^g) + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g). \tag{3}$$

DE/best/1

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g). \tag{4}$$

DE/best/2

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) + F \cdot (\mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g). \tag{5}$$

DE/rand/2

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g) + F \cdot (\mathbf{x}_{r4}^g - \mathbf{x}_{r5}^g). \tag{6}$$

It can be seen that the mutant vector $\mathbf{v}_i^g$ is generated by combing a base vector with one or two scaled difference vectors. In the above equations, the indices r_1, r_2, r_3, r_4, and r_5 are distinct integers randomly selected from [1,2, ... ,NP], and are all different from the index i. $\mathbf{x}_{\text{best}}^g$ is the vector with the best fitness value in the current generation. The factor F is a positive control parameter for weighting the difference vectors.

2) *Crossover:* In order to enhance population diversity, a crossover operation exchanges some components of the mutant vector $\mathbf{v}_i^g$ with the target vector $\mathbf{x}_i^g$ to generate a trial vector $\mathbf{u}_i^g$. The process can be expressed as

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, \text{ if rand}(0, 1) \leq CR \text{ or } j = j_{\text{rand}} \\ x_{i,j}^g, \text{ otherwise} \end{cases} \tag{7}$$

where rand(0,1) is a uniformly distributed random number as before. j_{rand} is an integer randomly generated from the range [1,D], which is used to ensure that the trial vector has at least one component different from the target vector. The crossover probability CR is another control parameter, which determines the fraction of vector components inherited from the mutant vector.

3) *Selection:* To decide whether the target or the trial vector can survive to the next generation, the selection operation is finally performed. For a minimization problem, the vector with the lower objective function value enters the next generation, which can be expressed as follows:

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, \text{ if } f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, \text{ otherwise} \end{cases} \tag{8}$$

where $f(\mathbf{x})$ is the objective function for the minimization problem.

B. Parameter Control Methods for DE

Control parameters in DE have significant effects on the performance of the algorithm [9], [16]. However, there is no fixed parameter setting that can achieve the best performance for all types of problems. Therefore, various parameter control methods have been proposed for DE to dynamically adjust the parameter values. These methods are capable of enhancing the robustness and efficiency of DE algorithm. In this paper, we classify the parameter control methods into two categories as follows.

1) *Population-Level Parameter Control:* All individuals in the population share the same parameter values that are controlled based on some deterministic rules or feedback information from the DE search process. Deterministic rules change the parameter values without exploiting any information from the evolution. Das *et al*. [18] proposed two schemes to control the scale factor F for DE. The first one decreases the value of F based on a linear rule, and the second one generates the value of F in a random way. Since the linear rule in the first scheme is based on both the current number and the predefined maximum number of generations, it is actually determined before running the algorithm.

By using some form of feedback from the DE search process, parameter control strategies dynamically adjust the parameter values that can adapt to different evolutionary states. In [19], the value of the parameter F is adaptively adjusted based on the minimum and maximum fitness values over the individuals in each generation. In [20], a fuzzy logic control approach was proposed to adapt the DE parameters F and CR. The fuzzy controllers incorporate the relative fitness values and individuals of the successive generations as their inputs, and the outputs are the values of F and CR. Zaharie [21] proposed a method of adapting the parameters of DE guided by the population diversity evolution. Based on the same idea, Zaharie and Petcu [22] further developed an adaptive Pareto DE for multiobjective optimization problems.

2) *Individual-Level Parameter Control:* In this form of adaptation, each individual in the population maintains its own set of parameter values, which are optimized through the evolutionary process. Brest *et al*. [24] introduced a self-adaptive approach for the control parameters F and CR. In each generation, new F_i and CR_i for each individual are randomly generated in their respective ranges with probabilities τ_1 and τ_2, respectively. Qin *et al*. [25] proposed a self-adaptive DE (SaDE) algorithm, in which the trial vector generation strategies and the control parameters F and CR are self-adapted by learning from the previous experiences. Zhang and

Sanderson [26] introduced a new adaptive DE algorithm called JADE. The control parameters for each individual in JADE are updated based on their historical record of success. In [27], JADE is further combined with a strategy adaptation mechanism. Wang *et al*. [28] proposed a composite DE (CoDE), which uses three trial vector generation strategies and three control parameter settings of F and CR. In each generation, each individual randomly combines these strategies and parameters to generate trial vectors. More recently, Zhong and Zhang [29] self-adapted the values of F and CR of DE for the subpixel mapping problem.

C. Variants of DE Mutation Strategies

Besides the five most frequently used mutation strategies mentioned in Section II-A, many other variants have been proposed to further improve the performance of DE. To make the performance of DE rotationally invariant, Price [32] proposed a new mutation strategy, called DE/current-to-rand/1. Price *et al*. [3] proposed the DE/rand/1/either-or algorithm, where the trial vectors that are either pure mutants or pure recombinants occur with probabilities p_F and $1 - p_F$, respectively. Fan and Lampinen [33] proposed a trigonometric mutation to increase the convergence speed of DE. Kaelo and Ali [34] proposed a hybrid mutation operator that uses the attraction-repulsion technique of an electromagnetism-like algorithm. More recently, Wang *et al*. [35] proposed a parameter-free Gaussian mutation strategy that uses the Gaussian sampling method to generate mutated individuals. This Gaussian mutation strategy was further hybridized with the classic DE/best/1 to balance the global search ability and convergence rate.

In order to balance the exploration and exploitation capabilities of DE, many researchers designed variants of some greedy mutation strategies [26], [36], [37]. These variants utilize the information of multiple good solutions instead of the best solution in the entire population. For example, DE/current-to-$p$best/1 in JADE [26] chooses any of the top $p\%$ solutions from the entire population to play the role of the single global best solution. Das *et al*. [36] proposed a local mutation model for DE/current-to-best/1, in which the best solution so far is chosen from a small neighborhood. The local mutation model is further combined with the global mutation model by a weight factor. Islam *et al*. [37] also proposed a variant of DE/current-to-best/1 that utilizes the best solution of a dynamic group of randomly selected solutions from the current population.

Besides developing new DE mutation strategies, some researchers investigated DE mutation frameworks that can be applied to different DE variants. For example, Epitropakis *et al*. [38] proposed a mutation framework in which the probability of selecting an individual to become a parent is inversely proportional to its distance from the individual undergoing mutation. Gong and Cai [39] proposed a kind of ranking-based mutation framework, where some of the parents in the mutation operator are proportionally selected according to their rankings in the current population. Cai and Wang [40] proposed a DE mutation framework that exploits the neighborhood and direction information of the population.

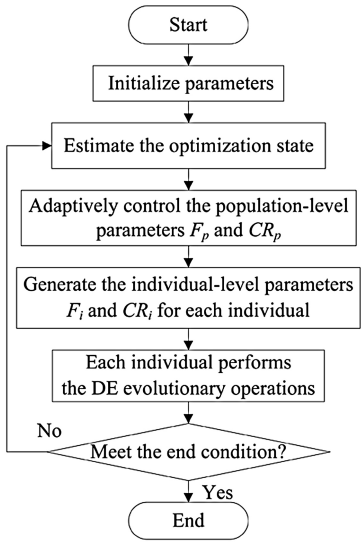


Fig. 1. Flowchart of the proposed ADE algorithm.

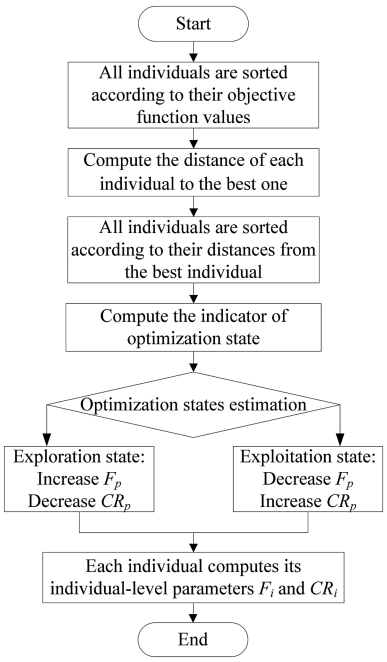


Fig. 2. Flowchart of the adaptive parameter control process.

III. Adaptive DE Algorithm

In this section, we propose a new ADE. The ADE is characterized by a new mutation strategy called DE/lbest/1 and a two-level adaptive parameter control scheme. The DE/lbest/1 mutation strategy helps balance the fast convergence and population diversity of the proposed algorithm. In the adaptive parameter control process, the optimization state is first estimated, and then the population-level parameter values F_p and CR_p are adjusted for the whole population. Finally, the individual-level parameter values F_i and CR_i for each individual are generated based on the values of F_p and CR_p. Fig. 1 illustrates the flowchart of the proposed ADE algorithm, and Fig. 2 shows the process of the two-level adaptive parameter control scheme.

A. DE/lbest/1 Mutation Strategy

For global numerical optimization, the experimental studies in [41] indicate that the DE/best/1 with binomial crossover is the most competitive approach among eight DE variants. DE/best/1 generates a mutant vector by combining the best vector with a scaled difference vector. Here, the best indicates the vector with the best fitness value in the entire population. In this way, all the vectors are guided by the same best solution information during the evolutionary search process. Such a greedy strategy is helpful for improving exploitation, which means the ability of a search algorithm to quickly converge to a near optimum. Nevertheless, due to the exploitative tendency, the population may lose its diversity too early and reduce its exploration ability to search new regions of the search space. Thus, individuals of the population are more likely to be trapped in some local optima, especially when solving complex multimodal problems. In order to cope with the problem of early loss of diversity and promote the exploration ability, we propose a new mutation strategy called DE/lbest/1.

In the proposed strategy, the entire population is divided into a predefined number of nonoverlapping groups simply based on the vector indices. Each group contains the same number of vectors. Since the vector indices are sorted randomly during initialization, the population is, in fact, randomly divided into equal-size groups. Note that the number of groups and group members are kept unchanged during the algorithm's execution. As mentioned earlier, a mutant vector in DE/best/1 is generated by combining a globally best vector with a scaled difference vector. In DE/lbest/1, the best vector is selected from the group to which the target vector belongs instead of the entire population. Therefore, the individuals of each group are attracted by the best vector of their own group, and the entire population is indeed guided by several locally best vectors instead of a single globally best vector. Such an approach thus benefits from a balance between fast convergence and population diversity. On the other hand, the difference vector involved in the mutation strategy can be generated not only by two vectors from the same group, but also by two vectors from different groups (i.e., the entire population). According to the above description, the DE/lbest/1 mutation strategy can be expressed as follows:

$$\mathbf{v}_i^g = \mathbf{x}_{\text{lbest}_i}^g + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) \tag{9}$$

where the subscript lbest_i denotes the best vector in the group with respect to target vector i, and r_1, r_2 are the difference vector indices that are indistinct integers randomly selected from [1, 2, ... , NP] and are also different from i. In the neighborhood-based local and global mutation models proposed for DE (DEGL) [36], also the local mutation model considers the vector index based on multiple groups of solutions to improve the exploration ability of DE. Our DE/lbest/1 differs from DEGL in the following major aspects.

1) DEGL employs both local and global mutation models, while our DE/lbest/1 utilizes only local mutation model (locally best solutions).
2) The local mutation model of DEGL considers the vector index based overlapping groups of solutions, while our DE/lbest/1 considers the vector index based nonoverlapping groups of solutions.
3) For DEGL, the locally best solution and two other solutions involved in the local mutation model are chosen from the same group of solutions. Differently, for DE/lbest/1, only the locally best solution is chosen from its respective group, while two other solutions are chosen from the entire population (may be from different groups).

B. Optimization State Estimation

In order to formulate an approach to optimization state estimation for DE, the population distribution characteristics are first described. At the early stage of evolution, the population distribution is relatively dispersive, since individuals are scattered in the searching space to explore different promising regions. As the optimization progresses, the population will gradually converge, and finally cluster around a global or local optimum at the later stage. Due to the variation, the information of population distribution can be used to estimate the optimization state in the DE algorithm. In the following paragraphs, we describe how to measure the population distribution and how to use the distribution information to estimate the optimization state.

In the procedure of optimization state estimation, all individuals are first sorted according to both their fitness values and their distances from the best individual. Furthermore, the relationship between these two sorting orders can be used to measure the population distribution. The detailed steps are as follows.

Step 1: In the beginning of each generation, the fitness values of all the individuals are sorted in a descending order (from the best to the worst). Suppose that the ranking of the fitness value of individual i is denoted as f_i, where $i = 1,2, \ldots, NP$.

Step 2: Compute the Euclidean distances from the best individual to the other individuals. Then, these distances are sorted in an ascending order (from the nearest to the farthest). Suppose that the ranking of the distance of individual i is denoted as d_i, where $i = 1,2, \ldots, NP$.

Step 3: After obtaining the two rankings f_i and d_i for each individual i, compute the indicator of the optimization state (IOS) so as to estimate the current optimization state

$$IOS = \sum_{i=1}^{NP} |f_i - d_i| . \tag{10}$$

If the two rankings for each individual are exactly the same (i.e., $f_i = d_i$ for each i), then the better individuals are also closer to the best individual, and IOS has its minimum value

$$IOS_{\min} = \sum_{r=1}^{NP} |r - r| = 0. \tag{11}$$

On the contrary, if the two rankings for each individual are just the opposite (i.e., $f_i + d_i = NP + 1$ for each i), then the better

individuals are also farther from the best individual, and IOS has its maximum value

$$IOS_{\max} = \begin{cases} \dfrac{NP \cdot NP}{2}, & \text{if } NP \text{ is even} \\ \dfrac{(NP+1) \cdot (NP-1)}{2}, & \text{if } NP \text{ is odd.} \end{cases} \tag{12}$$

Step 4: The value of IOS is normalized by the difference between the values of $IOS_{\max}$ and $IOS_{\min}$ as

$$\overline{IOS} = \frac{IOS - IOS_{\min}}{IOS_{\max} - IOS_{\min}} \tag{13}$$

where $\overline{IOS}$ is the normalized value of IOS, ranging from 0 to 1.

Step 5: Perform the estimation of the optimization state. According to the value of $\overline{IOS}$, we formally define the exploration and exploitation states based on

$$\Phi = \begin{cases} S_1, \text{ if rand}(0,1) < \overline{IOS} \\ S_2, \text{ otherwise} \end{cases} \tag{14}$$

where Φ is the estimated optimization state, S_1 and S_2 represent the exploration state and exploitation state, respectively, and rand(0,1) is a uniform random number within [0, 1].

Exploration State: If rand(0,1) is smaller than the value of $\overline{IOS}$, the optimization state is estimated to be the exploration state. That is, the optimization process has a probability of $\overline{IOS}$ to be classified into the exploration state.

Exploitation State: If rand(0,1) is not smaller than the value of $\overline{IOS}$, the optimization state is estimated to be the exploration state. That is, the optimization process has a probability of $(1 - \overline{IOS})$ to be classified into the exploitation state.

The definitions of exploration and exploitation states are based on the following considerations. Note that the definitions are based on considering the population set of the current generation. When the value of $\overline{IOS}$ is large, the differences between the two rankings f_i and d_i are obvious. A large $\overline{IOS}$ can be caused by two cases. For one case, there exist many good individuals far away from the best individual, which means that the population is exploring different promising regions. For another case, there exist many bad individuals close to the best, which indicates that the region around the best may be not the most promising one in the search space, because only a few good individuals have located there. Even if these bad individuals are close to the global optimum locating in a very narrow steep niche, there exist other better individuals that are farther from the optimum. These better individuals should be exploring other regions in the search space, or they would be closer to the optimum. In other words, only some but not most individuals have converged to the region around the global optimum. For both cases, the optimization process is more likely to be in the exploration state, since the population is indeed exploring different regions in this particular generation. This is also true if the population of previous generation favors more exploration (larger value of $\overline{IOS}$). Since the current and previous sets of populations are both exploring different regions in the search space, it is reasonable to individually estimate each as an exploration state (although to different degrees). On the contrary, when the value of $\overline{IOS}$ is small, most of the good individuals have converged around the best individual, and thus the optimization process has a large probability to be in the exploitation state.

C. Two-Level Adaptive Parameter Control Scheme

Our proposed adaptive parameter control scheme is a two-level adaptation strategy that includes two steps to adjust the parameters. In the first step, the population-level parameter values F_p and CR_p for the whole population are adaptively controlled. The control method is based on the optimization states that can be estimated by the approach described above. Based on the population-level parameter values F_p and CR_p, the individual-level parameter values F_i and CR_i for each individual are further computed in the second step. Since both the population and individual information are utilized in the control, the parameter values can be adapted to the characteristics of both population and individual.

Before describing the detail of the control scheme, we first briefly discuss the effects of the parameters F and CR. According to (9), the control parameter F is used to scale the difference vector. Using a large value of F generates a mutant vector largely different from the base vector chosen from the population, and thus helps maintain the population diversity. In contrast, a small value of F is more likely to facilitate convergence. According to (7), CR is the probability that a vector component will be inherited from the mutant vector. In our proposed mutation strategy, a mutant vector is generated by the guide of a locally best vector. Therefore, a large value of CR causes the newly generated vectors to converge around the locally best vectors, whereas a small value of CR makes the whole population more diverse.

Based on the above considerations, the strategies for adjusting the population-level parameter values F_p and CR_p in different optimization states are defined as follows.

1) *Exploration State—Increasing F_p and Decreasing CR_p:* In the exploration state, in order to explore more promising regions, it is better to increase the value of F_p. Conversely, the value of CR_p should be decreased so that the newly generated vectors will not crowd around some locally best vectors.

2) *Exploitation State—Decreasing F_p and Increasing CR_p:* In the exploitation state, an appropriate way to accelerate convergence is to decrease the value of F_p. Meanwhile, increasing the value of CR_p can help exploitation around some promising vectors.

Based on the above strategies, the values of F_p and CR_p can be adjusted adaptively according to the current estimated optimization state Φ. The adjustment is based on the values of F_p and CR_p of the previous generation, as shown in (15) and (16)

$$F_p^g = \begin{cases} F_p^{g-1} + c_F \cdot \Delta F_p, \text{ if } \Phi = S_1 \\ F_p^{g-1} - c_F \cdot \Delta F_p, \text{ if } \Phi = S_2 \end{cases} \tag{15}$$

$$CR_p^g = \begin{cases} CR_p^{g-1} - c_{CR} \cdot \Delta CR_p, \text{ if } \Phi = S_1 \\ CR_p^{g-1} + c_{CR} \cdot \Delta CR_p, \text{ if } \Phi = S_2 \end{cases} \tag{16}$$

where

$$\Delta F_p, \Delta CR_p = \begin{cases} \dfrac{IOS - IOS_{\min}}{IOS_{\max} - IOS_{\min}}, \text{if}\,\Phi = S_1 \\ \dfrac{IOS_{\max} - IOS}{IOS_{\max} - IOS_{\min}}, \text{if}\,\Phi = S_2. \end{cases} \quad (17)$$

Obviously, the values of ΔF_p and ΔCR_p are clamped in the range of [0,1]. The coefficients c_F and c_{CR} are used to control the adjustment steps $c_F \cdot \Delta F_p$ and $c_{CR} \cdot \Delta CR_p$, respectively. In order to shrink the adjustment steps smaller than 10% of the value ranges of F and CR (i.e., 0.1), we set c_F and c_{CR} as 0.1 and 0.05 empirically. According to (17), the adjustment step is also related to the value of IOS. The values of F_p and CR_p are both clamped in the range of [0,1].

After obtaining the population-level parameter values F_p and CR_p for the whole population, the individual-level parameter values F_i and CR_i are generated for each individual i according to their rankings of fitness values and the distances from the best individual. It is implemented by adjusting the values of F_p and CR_p of the current generation

$$F_i^g = \begin{cases} F_p^g + \Delta F_i, \text{if}(f_i > \frac{NP}{2}) \wedge (d_i > \frac{NP}{2}) \\ F_p^g - \Delta F_i, \text{if}(f_i < \frac{NP}{2}) \wedge (d_i < \frac{NP}{2}) \\ F_p^g, \text{otherwise} \end{cases} \quad (18)$$

$$CR_i^g = \begin{cases} CR_p^g - \Delta CR_i, \text{if}(f_i > \frac{NP}{2}) \wedge (d_i > \frac{NP}{2}) \\ CR_p^g + \Delta CR_i, \text{if}(f_i < \frac{NP}{2}) \wedge (d_i < \frac{NP}{2}) \\ CR_p^g, \text{otherwise} \end{cases} \quad (19)$$

where

$$\Delta F_i, \Delta CR_i = \begin{cases} \frac{(f_i + d_i) - NP}{2NP}, \text{if}(f_i > \frac{NP}{2}) \wedge (d_i > \frac{NP}{2}) \\ \frac{NP - (f_i + d_i)}{2NP}, \text{if}(f_i < \frac{NP}{2}) \wedge (d_i < \frac{NP}{2}). \end{cases} \quad (20)$$

In these adaptive strategies for the individual-level parameters, we can see that if individual i has a low fitness value and is far from the best individual, a large F_i and a small CR_i can help it to explore new promising regions of the search space. In this case, F_i and CR_i should be generated by increasing F_p and decreasing CR_p, respectively. In contrast, if individual i has a high fitness value and is near the best individual, a small F_i and a large CR_i would allow it to do more exploitation around its current position. Thus, F_i and CR_i should be generated by decreasing F_p and increasing CR_p, respectively. Based on the above considerations, only the individuals with both rankings either high or low will adjust the parameter values accordingly, while the rest of individuals utilize only the population-level parameters. In this implementation, the rankings above or below half of the population size are considered high or low, respectively. The values of F_i and CR_i are also clamped in the range of [0,1].

D. Runtime Complexity of ADE

The runtime complexity of a classic DE algorithm is $O(NP \cdot D \cdot G_{\max})$ [36], where $G_{\max}$ is the maximum number of generations. ADE differs from the classic DE mainly in the DE/lbest/1 mutation strategy and the two-level parameter adaptation.

For the DE/lbest/1, the runtime complexity of finding the locally best vector in each group depends on comparing the fitness value with that of the locally best vector. Note that the locally best fitness value should be upgraded for each target vector in the respective group if it is replaced by the newly generated trial vector. In the worst possible case, when the trial vector always replaces the target vector, the overall runtime complexity is $O(NP \cdot D \cdot G_{\max})$.

For the population-level adaptation, we need to sort all the vectors based on their fitness values and their distances to the best vector, respectively, in the beginning of each generation. The runtime complexity of computing the distances for all the vectors is $O(NP \cdot D)$ since we use Euclidean distance. By utilizing the heap sort algorithm, the sorting procedure can be completed in $O(NP \cdot \log_2 NP)$ time. For the individual-level adaptation, each vector needs to compare their rankings against $\frac{NP}{2}$ [see (18) and (19)], and the number of additional comparisons is $O(NP)$. Hence, over $G_{\max}$ generations, the overall runtime complexity of the two-level parameter adaptation is $O(\max(NP \cdot D \cdot G_{\max}, NP \cdot \log_2 NP \cdot G_{\max}))$.

Considering both DE/lbest/1 and two-level parameter adaptation, the runtime complexity of ADE is $O(\max(NP \cdot D \cdot G_{\max}, NP \cdot \log_2 NP \cdot G_{\max}))$. If $D \geq \log_2 NP$, the asymptotic order of complexity for ADE remains $O(NP \cdot D \cdot G_{\max})$. This condition is usually satisfied when D is relatively high. In this case, ADE does not impose any serious burden on the runtime complexity.

IV. Experimental Results

A. Benchmark Functions and Experimental Setup

In this section, experiments are carried out to evaluate the performance of the proposed ADE. We use 33 benchmark functions chosen from [42] and [43]. Functions f_1–f_{22} are summarized in Table I [42]. A detailed description of functions f_{23}–f_{33} can be found in [43]. These functions can be classified into four groups. The first six functions f_1–f_6 are unimodal functions. These functions can be used to test the convergence speed of the algorithms. The next six functions f_7–f_{12} are multimodal functions where the number of local optima increases exponentially with the problem dimension. The next ten functions f_{13}–f_{22} are low-dimensional multimodal functions with a few local optima. The algorithms' global search ability to escape from local optima can be verified by these multimodal functions. The last 11 functions f_{23}–f_{33} are hybrid composition functions taken from the CEC 2005 Competition (F_{15}–F_{25}) [43], each of which is composed of ten subfunctions. Obviously, the functions in the fourth group are much more complex and make our test suite more comprehensive and convincing.

The population size NP of ADE is set to 50 and 200 for problems with $D \leq 30$ and $D = 100$, respectively. The population of ADE is divided into ten groups. The initial values of F_p and CR_p of ADE are both set as 0.5. Each experiment is run 25 times independently and the results are averaged. In addition, we make use of the Wilcoxon's rank sum test [44] at $\alpha = 0.05$ to evaluate the statistical significance of the results.

TABLE I
BENCHMARK FUNCTIONS [42], [43]

Name	Test function	D	S	f_{min}
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	30, 100	$[-100, 100]^D$	0
Schewefel 2.22	$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30, 100	$[-10, 10]^D$	0
Schewefel 1.2	$f_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	30, 100	$[-100, 100]^D$	0
Rosenbrock	$f_4(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30, 100	$[-30, 30]^D$	0
Step	$f_5(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	30, 100	$[-100, 100]^D$	0
Noisy Quaric	$f_6(x)=\sum_{i=1}^{D} ix_i^4+random[0,1)$	30, 100	$[-1.28, 1.28]^D$	0
Schewefel 2.26	$f_7(x)=\sum_{i=1}^{D}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30, 100	$[-500, 500]^D$	-12569.5 for D=30
Rastrigin	$f_8(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	30, 100	$[-5.12, 5.12]^D$	0
Ackley	$f_9(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i)+20+e$	30, 100	$[-32, 32]^D$	0
Griewank	$f_{10}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	30, 100	$[-600, 600]^D$	0
Penalized	$f_{11}(x)=\frac{\pi}{D}\{10\sin^2(\pi y_i)+\sum_{i-1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\}+\sum_{i=1}^{D}u(x_i,10,100,4)$	30, 100	$[-50, 50]^D$	0
	$f_{12}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin 2(3\pi x_{i+1})]+(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^{D}u(x_i,5,100,4)$	30, 100	$[-50, 50]^D$	0
Foxholes	$f_{13}(x)=(1/500+\sum_{j=1}^{25}(j+\sum_{i=1}^{2}(x_i-a_{ij})^6)^{-1})^{-1}$	2	$[-65.536, 65.536]^D$	1
Kowalik	$f_{14}(x)=\sum_{i=1}^{11}[a_i-x_1(b_i^2+b_ix_2)/(b_i^2+b_ix_3+x_4)]^2$	4	$[-5, 5]^D$	0.0003075
Six-hump Camel-back	$f_{15}(x)=4x_1^2-2.1x_1^2+3^{-1}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5, 5]^D$	-1.0316285
Branin	$f_{16}(x)=(x_2-5.1/4\pi^2\cdot x_1^2+5/\pi\cdot x_1-6)^2+10(1-1/8\pi)\cos x_1+10$	2	[-5, 10] x [0, 15]	0.397887
Goldstein-price	$f_{17}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2, 2]^D$	3.00000
Hartman	$f_{18}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{4}a_{ij}(x_j-p_{ij})^2)$	4	$[0, 1]^D$	-3.86278
	$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	6	$[0, 1]^D$	-3.32237
Shekel's	$f_{20}(\mathbf{x})=-\sum_{i=1}^{5}((\mathbf{x}-\mathbf{a}_i)(\mathbf{x}-\mathbf{a}_i)^T+c_i)^{-1}$	4	$[0, 10]^D$	-10.1532
	$f_{21}(\mathbf{x})=-\sum_{i=1}^{7}((\mathbf{x}-\mathbf{a}_i)(\mathbf{x}-\mathbf{a}_i)^T+c_i)^{-1}$	4	$[0, 10]^D$	-10.4029
	$f_{22}(\mathbf{x})=-\sum_{i=1}^{10}((\mathbf{x}-\mathbf{a}_i)(\mathbf{x}-\mathbf{a}_i)^T+c_i)^{-1}$	4	$[0, 10]^D$	-10.5364
Hybrid composition functions	f_{23}-f_{33}: F_{15}-F_{25} from the IEEE CEC 2005 special session and competition [43]	30	given in [43]	given in [43]

TABLE II
COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON UNIMODAL FUNCTIONS

Fun.	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
f_1	150000	**1.49E-70 (3.95E-70)**	4.85E-25† (6.07E-25)	2.19E-57† (1.10E-56)	4.67E-31† (5.08E–31)	5.02E-65† (5.96E-65)
f_2	200000	**3.21E-51 (6.57E-51)**	2.98E-19† (1.77E-19)	1.07E-27† (5.32E–27)	1.22E-25† (1.24E–25)	6.28E-46† (6.09E–46)
f_3	500000	8.13E-27 (2.60E-26)	8.40E-05† (8.25E-05)	**3.55E-99† (1.68E–98)**	1.71E-14† (5.81E–14)	3.22E-28† (1.03E–28)
f_4	2000000	2.28E-29 (3.84E-29)	**0.00E+00† (0.00E+00)**	3.19E-01† (1.10E+00)	3.19E-01† (1.08E+00)	3.19E-01† (1.10E+00)
f_5	150000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_6	300000	2.88E-03 (1.17E-03)	3.62E-03† (1.22E-03)	**5.94E-04† (2.43E–04)**	3.40E-03† (7.71E–04)	1.34E-03† (4.43E–04)

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

TABLE III
SUCCESS RATE AND SEARCH SPEED COMPARISONS ON UNIMODAL FUNCTIONS

Fun.	Acceptable accuracy		ADE	CoDE	JADE	jDE	SaDE
f_1	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	**2.89E+04**	7.58E+04	3.42E+04	6.41E+04	3.18E+04
		rank	**1**	5	3	4	2
f_2	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	**4.60E+04**	1.18E+05	6.35E+04	9.07E+04	5.13E+04
		rank	**1**	5	3	4	2
f_3	1E-10	SR	**100**	0	**100**	**100**	**100**
		FEs	2.30E+05	N/A	**7.74E+04**	3.98E+05	2.15E+05
		rank	3	5	**1**	4	2
f_4	1E-10	SR	**100**	**100**	92	92	92
		FEs	2.73E+05	4.98E+05	**9.96E+04**	6.76E+05	3.18E+05
		rank	**1**	2	3	5	4
f_5	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	1.12E+02	2.14E+02	1.00E+02	2.10E+02	**9.20E+01**
		rank	3	5	2	4	**1**
f_6	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	7.76E+04	1.18E+05	**3.06E+04**	1.04E+05	3.61E+04
		rank	3	5	**1**	4	2
avg_rank			**2**	4.5	2.2	4.2	2.2

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

We compare ADE with four state-of-the-art DE variants, i.e., CoDE [28], JADE [26], jDE [24], and SaDE [25] on all test functions with 30 dimensions (except the low-dimensional functions f_{13}–f_{22}). All these algorithms use some strategies to adaptively control the parameters F and CR. The other parameters of the four algorithms are set according to their original papers.

1) *Unimodal Functions:* The mean and standard deviation of the results for 25 independent runs on f_1–f_6 are presented in Table II. Each algorithm is run for the number of fitness evaluations indicated in the FEs column of the table [42]. For clarity, the results of the best algorithms are marked in boldface. According to the results, it can be seen that ADE, JADE, and SaDE perform better than CoDE and jDE on these functions in general. This is because the former three algorithms use greedy mutation strategies to some degree. The relatively low accuracy results of CoDE and jDE should be due to their more diverse mutation strategies. It is interesting to note that CoDE achieves the best results on f_4, which is a unimodal function for $D \leq 3$, but may become multimodal when the dimension is high [45].

In order to compare the reliability and search speed of different algorithms, Table III summarizes the success rates at which acceptable solutions are found over 25 runs and the number of FEs required to find the acceptable solutions. A solution is considered acceptable if it is obtained with sufficient accuracy: 1E-10. The ranks in the table are evaluated based on the descending order of the success rates and ties are broken by the ascending order of FEs. From Table III, we find that all the algorithms perform with a high reliability on most of the unimodal functions. In particular, the proposed ADE algorithm is able to find acceptable solutions with 100% success rate on all the functions. On the other hand, only CoDE and ADE always obtain the near-global optimum for f_4, while JADE, jDE, and SaDE sometimes fail to locate the acceptable solutions. However, CoDE fully fails to reach acceptable solutions for f_3. From the aspect of search speed, JADE, SaDE, and ADE use a smaller number of FEs than CoDE and jDE do to reach acceptable results on average. Overall, ADE performs the best on most functions in this group.

2) *Multimodal Functions With Many Local Optima:* Table IV compares the five DE variants on the multimodal functions with many local optima. It can be seen that ADE, CoDE, and jDE are capable of reaching near-global optima on all the six functions. These three algorithms obtain the same best results on f_7, f_8, and f_{10}, but CoDE and jDE get higher accuracy than ADE on f_9, while ADE outperforms CoDE and jDE on f_{11} and f_{12}. On the other hand, JADE

TABLE IV

COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON MULTIMODAL FUNCTIONS WITH MANY LOCAL OPTIMA

Fun.	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
f_7	900000	**-12569.49 (1.82E-12)**	**-12569.49 (1.86E-12)**	-12564.75 (2.37E+01)	**-12569.49 (7.28E-12)**	-12550.54† (4.43E+01)
f_8	500000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.63E+00† (1.37E+00)
f_9	200000	5.99E-15 (1.77E-15)	**4.14E-15† (0.00E+00)**	**4.14E-15† (0.00E+00)**	4.28E-15† (6.96E-16)	**4.14E-15† (0.00E+00)**
f_{10}	200000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	3.94E-04† (1.97E-03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{11}	150000	**1.57E-32 (2.74E-48)**	2.31E-26† (2.73E-26)	**1.57E-32 (2.79E-48)**	3.79E-32† (1.91E-32)	**1.57E-32 (2.79E-48)**
f_{12}	150000	**1.35E-32 (5.47E-48)**	1.91E-25† (2.51E-25)	**1.35E-32 (5.59E-48)**	2.17E-31† (2.41E-31)	**1.35E-32 (5.59E-48)**

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

TABLE V

SUCCESS RATE AND SEARCH SPEED COMPARISONS ON MULTIMODAL FUNCTIONS WITH MANY LOCAL OPTIMA

Fun.	Acceptable accuracy		ADE	CoDE	JADE	jDE	SaDE
f_7	-10000	SR	**100**	**100**	**100**	**100**	**100**
		FEs	2.42E+04	2.43E+04	**2.21E+04**	3.36E+04	3.95E+04
		rank	2	3	**1**	4	5
f_8	1E-10	SR	**100**	**100**	**100**	**100**	28
		FEs	1.74E+05	1.97E+05	1.45E+05	**1.23E+05**	1.60E+05
		rank	3	4	2	**1**	5
f_9	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	**4.93E+04**	1.22E+05	5.52E+04	1.10E+05	5.14E+04
		rank	**1**	5	3	4	2
f_{10}	1E-10	SR	**100**	**100**	96	**100**	**100**
		FEs	5.84E+04	9.03E+04	3.73E+04	6.79E+04	**3.53E+04**
		rank	2	4	5	3	**1**
f_{11}	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	5.53E+04	6.90E+04	3.39E+04	5.92E+04	**2.91E+04**
		rank	3	5	2	4	**1**
f_{12}	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	3.93E+04	7.28E+04	3.59E+04	6.32E+04	**3.11E+04**
		rank	3	5	2	4	**1**
avg_rank			**2.3**	4.3	2.5	3.3	2.5

and SaDE perform the best on f_9, f_{11}, and f_{12}, but they are sometimes trapped in local optima on f_{10} and f_8, respectively. These results indicate that ADE manages to avoid premature convergence and also finds solutions with high accuracy on multimodal functions with a lot of local optima.

The reliability and search speed of different DE variants are further compared and reported in Table V. The acceptable accuracy is set as $-10\,000$ for f_7 and 1E-10 for all others. The results show that all six functions are optimized by ADE, CoDE, and jDE with 100% success rates. Moreover, the proposed ADE reaches the acceptable solutions with a smaller number of FEs than CoDE and jDE. Although JADE and SaDE converge faster than ADE on some functions, their reliability is relatively low, as reflected by their success rates on f_{10} and f_8, respectively. Overall, the ADE also exhibits very competitive performance on multimodal functions with a lot of local optima. Not only does it benefit from fast convergence but also high algorithm reliability that may stem from the two-level adaptive parameter control scheme.

3) *Multimodal Functions With Few Local Optima:* For multimodal functions with a few local optima, we run the algorithms for a small number of FEs to find out whether they can locate the global optimum quickly. The experimental results are tabulated in Table VI. We find that ADE and SaDE are the best among the five DE variants for this group of functions. They are able to find the global optimum within the predefined maximum number of FEs for all the functions. CoDE, JADE, and jDE perform worse than ADE and SaDE on one or more functions, e.g., f_{14}, f_{19}, and f_{20}. Overall, the performance of different algorithms is similar on most of functions in this group, and we skip the comparison of success rate and number of FEs. ADE remains the best one among these algorithms.

4) *Hybrid Composition Functions:* All of the functions in this group are nonseparable, rotated, and multimodal functions with a huge number of local optima. Such functions appear to be the most difficult class of problems for most optimization algorithms. The experimental results are shown in Tables VII and VIII. It can be noted that no DE variant is able to obtain a near-global optimum on any function. Overall, ADE achieves better results on more functions. It performs better than CoDE, JADE, jDE, and SaDE on five, seven, eight, and eight functions, respectively. Conversely, CoDE, JADE, and jDE surpass ADE on three, one, and two functions, respectively. SaDE cannot outperform ADE on any functions in this group.

In terms of reliability, ADE achieves 100% success rates on all these functions except f_{23}, while the other algorithms

TABLE VI
COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON MULTIMODAL FUNCTIONS WITH A FEW LOCAL OPTIMA

Fun.	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
f_{13}	20000	**0.998004 (0.00E+00)**	**0.998004 (0.00E+00)**	**0.998004 (1.36E-16)**	**0.998004 (0.00E+00)**	**0.998004 (0.00E+00)**
f_{14}	20000	**0.0003075 (4.03E-13)**	**0.0003075 (2.56E-08)**	0.0011099† (4.01E-03)	0.0003544† (1.82E-04)	**0.0003075 (1.02E-12)**
f_{15}	20000	**-1.0316285 (4.44E-16)**	**-1.0316285 (4.53E-16)**	**-1.0316285 (2.82E-14)**	**-1.0316285 (4.44E-16)**	**-1.0316285 (4.53E-16)**
f_{16}	20000	**0.397887 (0.00E+00)**	**0.397887 (0.00E+00)**	**0.397887 (7.73E-11)**	**0.397887 (0.00E+00)**	**0.397887 (0.00E+00)**
f_{17}	20000	**3.000000 (1.78E-15)**	**3.000000 (1.81E-15)**	**3.000000 (1.50E-15)**	**3.000000 (1.35E-15)**	**3.000000 (1.81E-15)**
f_{18}	20000	**-3.86274 (8.33E-16)**	**-3.86274 (8.65E-16)**	**-3.86274 (9.06E-16)**	**-3.86274 (8.88E-16)**	**-3.86274 (8.21E-16)**
f_{19}	20000	**-3.321995 (1.26E-15)**	-3.302972 (4.45E-02)	-3.288704† (5.45E-02)	-3.307725† (3.86E-02)	**-3.321995 (1.84E-09)**
f_{20}	20000	**-10.1532 (4.19E-15)**	**-10.1532 (3.80E-15)**	-9.9511 (1.01E+00)	**-10.1532 (4.74E-11)**	**-10.1532 (4.93E-15)**
f_{21}	20000	**-10.4029 (4.11E-15)**	**-10.4029 (3.52E-15)**	**-10.4029 (6.36E-10)**	**-10.4029 (3.47E-14)**	**-10.4029 (3.46E-15)**
f_{22}	20000	**-10.5364 (2.71E-15)**	**-10.5364 (2.46E-15)**	**-10.5364 (1.16E-10)**	**-10.5364 (2.11E-14)**	**-10.5364 (3.40E-15)**

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

TABLE VII
COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON HYBRID COMPOSITION FUNCTIONS

Fun.	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
f_{23}	300000	**1.80E+02 (1.31E+02)**	4.16E+02† (6.25E+01)	3.88E+02† (6.52E+01)	3.28E+02† (1.08E+02)	3.88E+02† (5.15E+01)
f_{24}	300000	1.20E+02 (3.27E+01)	**7.95E+01† (6.97E+01)**	1.20E+02 (1.41E+02)	8.97E+01† (6.40E+01)	1.32E+02 (2.13E+01)
f_{25}	300000	1.27E+02 (2.92E+01)	**7.86E+01† (7.68E+01)**	1.56E+02 (1.39E+02)	1.42E+02 (3.12E+01)	1.71E+02† (1.95E+01)
f_{26}	300000	**8.58E+02 (1.08E+02)**	9.05E+02† (1.27E+00)	9.04E+02† (7.30E-01)	9.04E+02† (8.16E-01)	8.76E+02 (5.26E+01)
f_{27}	300000	**8.70E+02 (5.75E+01)**	9.04E+02† (6.11E-01)	9.04E+02† (1.05E+00)	9.04E+02† (3.92E-01)	8.73E+02 (5.47E+01)
f_{28}	300000	**8.65E+02 (5.76E+01)**	9.04E+02† (1.06E+00)	9.04E+02† (9.69E-01)	9.04E+02† (9.71E-01)	8.75E+02 (5.18E+01)
f_{29}	300000	**5.00E+02 (1.31E-13)**	**5.00E+02 (9.85E-14)**	**5.00E+02 (1.89E-13)**	**5.00E+02 (1.62E-13)**	5.12E+02 (5.88E+01)
f_{30}	300000	9.25E+02 (1.22E+01)	**8.55E+02† (2.28E+01)**	8.56E+02† (2.24E+01)	8.62E+02† (2.09E+01)	9.25E+02 (1.29E+01)
f_{31}	300000	**5.34E+02 (5.45E-03)**	**5.34E+02 (4.09E-04)**	5.50E+02 (7.89E+01)	5.40E+02 (2.96E+01)	**5.34E+02 (5.99E-03)**
f_{32}	300000	**2.00E+02 (1.56E-12)**	**2.00E+02 (2.90E-14)**	**2.00E+02 (2.84E-14)**	8.60E+02† (2.88E+02)	**2.00E+02 (2.84E-14)**
f_{33}	300000	**2.10E+02 (4.10E-01)**	2.11E+02† (7.32E-01)	2.11E+02† (5.88E-01)	2.11E+02† (6.47E-01)	2.12E+02† (1.25E+00)

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

yield relatively low success rates on three or four functions. In terms of search speed, JADE converges fastest on most functions and ADE is the second fastest. ADE still obtains the smallest value of average rank on this group of functions. Although the average rank of JADE is very close to that of ADE, the former performs the worst in terms of reliability on three functions.

The convergence curves of CoDE, JADE, jDE, SaDE, and ADE are plotted in Fig. 3 for some selected benchmark functions.

C. Scalability of ADE

In order to demonstrate the scalability of ADE, we further compare the performance of the five DE variants on f_1–f_{12} with $D = 100$. Table IX reports the mean and standard deviation of the results. For most unimodal functions, the performance of ADE, JADE, and SaDE remains significantly superior to that of CoDE and jDE. One may note that the performance of CoDE deteriorates with the increase of dimension for f_4, where CoDE achieves the best results for $D = 30$. For the multimodal functions, we can observe that the algorithms obtain similar rankings as on 30D functions. Overall, ADE, CoDE, and jDE present better global search ability than JADE and SaDE. ADE outperforms all other algorithms on f_7, f_8, and f_{12}.

Reliability and search speed comparisons are shown in Table X. ADE and JADE yield 100% success rates on all test functions, while CoDE, jDE, and SaDE achieve relatively low success rates on two, two, and three functions, respectively. From the aspect of search speed, SaDE converges fastest on most functions, followed by ADE. CoDE and JADE exhibit similar convergence performance while jDE converges slowest in general. These results demonstrate that ADE remains good at striking a balance between reliability and search speed when the dimension increases to 100.

D. Effects of ADE Components

Two main components of ADE are the DE/lbest/1 mutation strategy and the two-level parameter adaptation. First, we compare DE using DE/best/1 and DE using DE/lbest/1 to reveal the effectiveness of the DE/lbest/1 mutation strategy. They use the same parameter setting of $NP = 50$, $F = 0.5$, and $CR = 0.5$. The population of DE/lbest/1 is divided into ten groups. The results on f_1–f_{12} with $D = 30$ are shown in Table XI. We can observe that DE/best/1 can find very high accuracy solutions on simple unimodal functions f_1–f_3, but it fails to locate the near-global optima on most multimodal functions. The proposed DE/lbest/1 is helpful for balancing the performance between unimodal and multimodal functions. It can find the near-global optimum on most multimodal functions while still being able to converge quickly on unimodal functions. The strengths of the DE/lbest/1 are further demonstrated by comparing with DE/current-to-$p$best/1 in JADE ($F = 0.5$, $CR = 0.5$), which also utilizes multiple good solutions. The results of DE/current-to-$p$best/1 on f_1–f_{12} can be found in the last column of Table XI. We can observe that the overall performance of DE/lbest/1 is

TABLE VIII
SUCCESS RATE AND SEARCH SPEED COMPARISONS ON HYBRID COMPOSITION FUNCTIONS

Fun.	Acceptable accuracy		ADE	CoDE	JADE	jDE	SaDE
f_{23}	2E+02	SR	**40**	4	0	**40**	4
		FEs	**3.86E+04**	5.05E+04	N/A	7.46E+04	5.41E+04
		rank	**1**	3	5	2	4
f_{24}	2E+02	SR	**100**	96	80	96	96
		FEs	**1.94E+04**	3.11E+04	3.32E+04	4.94E+04	5.11E+04
		rank	**1**	2	5	3	4
f_{25}	2E+02	SR	**100**	96	80	92	92
		FEs	5.49E+04	5.82E+04	**1.06E+04**	9.89E+04	1.44E+05
		rank	**1**	2	5	3	4
f_{26}	1E+03	SR	**100**	**100**	**100**	**100**	**100**
		FEs	4.30E+03	4.41E+03	**2.03E+03**	4.50E+03	4.80E+03
		rank	2	3	**1**	4	5
f_{27}	1E+03	SR	**100**	**100**	**100**	**100**	**100**
		FEs	4.30E+03	4.32E+03	**2.08E+03**	4.55E+03	4.65E+03
		rank	2	3	**1**	4	5
f_{28}	1E+03	SR	**100**	**100**	**100**	**100**	**100**
		FEs	4.25E+03	4.41E+03	**2.06E+03**	4.82E+03	4.65E+03
		rank	2	3	**1**	5	4
f_{29}	1E+03	SR	**100**	**100**	**100**	**100**	28
		FEs	3.70E+03	4.95E+03	**2.88E+03**	7.74E+03	4.25E+03
		rank	2	4	**1**	5	3
f_{30}	1E+03	SR	**100**	**100**	**100**	**100**	**100**
		FEs	2.49E+04	1.04E+04	**5.70E+03**	1.21E+04	1.53E+04
		rank	5	2	**1**	3	4
f_{31}	1E+03	SR	**100**	**100**	**100**	**100**	**100**
		FEs	3.85E+03	5.22E+03	**2.94E+03**	7.91E+03	4.40E+03
		rank	2	4	**1**	5	3
f_{32}	5E+02	SR	**100**	**100**	**100**	16	**100**
		FEs	5.65E+03	8.46E+03	**4.73E+03**	1.36E+04	5.70E+03
		rank	2	4	**1**	5	3
f_{33}	5E+02	SR	**100**	**100**	**100**	**100**	**100**
		FEs	4.35E+03	6.63E+03	**3.47E+03**	8.51E+03	5.00E+03
		rank	2	4	**1**	5	3
avg_rank			**2**	3.1	2.1	4.0	3.8

TABLE IX
COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON 100 *D* FUNCTIONS

Fun.	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
f_1	1000000	3.80E-100 (7.01E-100)	1.01E-72† (8.70E-73)	5.06E-76† (6.63E-76)	3.14E-22† (2.98E–22)	**1.84E-137† (3.32E-137)**
f_2	1000000	6.57E-47 (3.17E-46)	1.10E-40† (8.11E-41)	5.17E-42† (7.91E–42)	1.93E-13† (4.41E–14)	**1.19E-83† (2.78E–83)**
f_3	1000000	3.92E-03 (2.47E-03)	9.29E+02† (3.01E+02)	**9.99E-09† (1.05E–08)**	3.68E+02† (1.14E+02)	2.27E-02† (2.07E–02)
f_4	1000000	**2.98E+00 (6.45E+00)**	6.27E+01† (3.06E+01)	5.25E+00† (6.69E+00)	9.56E+01† (1.91E+01)	7.10E+01† (3.75E+01)
f_5	1000000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_6	1000000	3.31E-02 (7.27E-03)	1.25E-02† (3.60E-03)	**1.88E-03† (4.08E–04)**	1.98E-02† (2.50E–03)	1.27E-02† (2.45E–03)
f_7	1000000	**-41898.29 (1.95E-11)**	-41893.55 (2.37E+01)	-41879.02† (5.46E+00)	-41897.95† (5.16E-01)	-39555.56† (5.43E+02)
f_8	1000000	**6.37E-01 (1.97E+00)**	1.04E+01† (3.25E+00)	7.53E+00† (6.72E-01)	1.99E+00† (1.81E+00)	3.65E+01† (5.23E+00)
f_9	1000000	2.72E+00 (4.89E-01)	**4.85E-15† (1.45E-15)**	1.14E-14† (1.25E-15)	5.97E-12† (2.65E-12)	1.56E+00† (2.81E-01)
f_{10}	1000000	1.28E-03 (3.49E-03)	3.94E-04 (1.97E-03)	1.28E-03 (3.56E-03)	**0.00E+00 (0.00E+00)**	9.04E-03† (1.59E-02)
f_{11}	1000000	3.12E-02 (6.95E-02)	**4.71E-33† (1.40E-48)**	4.61E-02† (8.58E-02)	1.05E-22 (5.83E-23)	2.49E-02 (3.70E-02)
f_{12}	1000000	**1.75E-32 (4.65E-33)**	4.39E-04† (2.20E-03)	4.39E-04† (2.20E-03)	1.06E-21† (6.40E-22)	6.14E-01† (1.23E+00)

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

better than that of DE/current-to-*p*best/1. In fact, DE/lbest/1 performs better than or at least the same as DE/current-to-*p*best/1 on nine out of the twelve test functions, while DE/current-to-*p*best/1 beats DE/lbest/1 on three test functions.

In order to investigate the effect of the population-level parameter control, we consider ADE without parameter control (i.e., DE/lbest/1) and ADE with population-level control only (denoted as ADE-ponly). Three DE/lbest/1 algorithms with different parameter settings are compared with ADE-ponly on f_1–f_{12}. All of them set F to 0.5 as suggested in most of the literature [2], [16], and set CR to 0.1, 0.5, and 0.9, respectively. The experimental results averaged over 25 runs are listed in the last four columns of Table XII. Although DE/lbest/1 performs better than DE/best/1, the performance of DE/lbest/1 is still sensitive to the parameter settings. The DE/lbest/1 ($CR = 0.1$) is able to find the near-global optimum

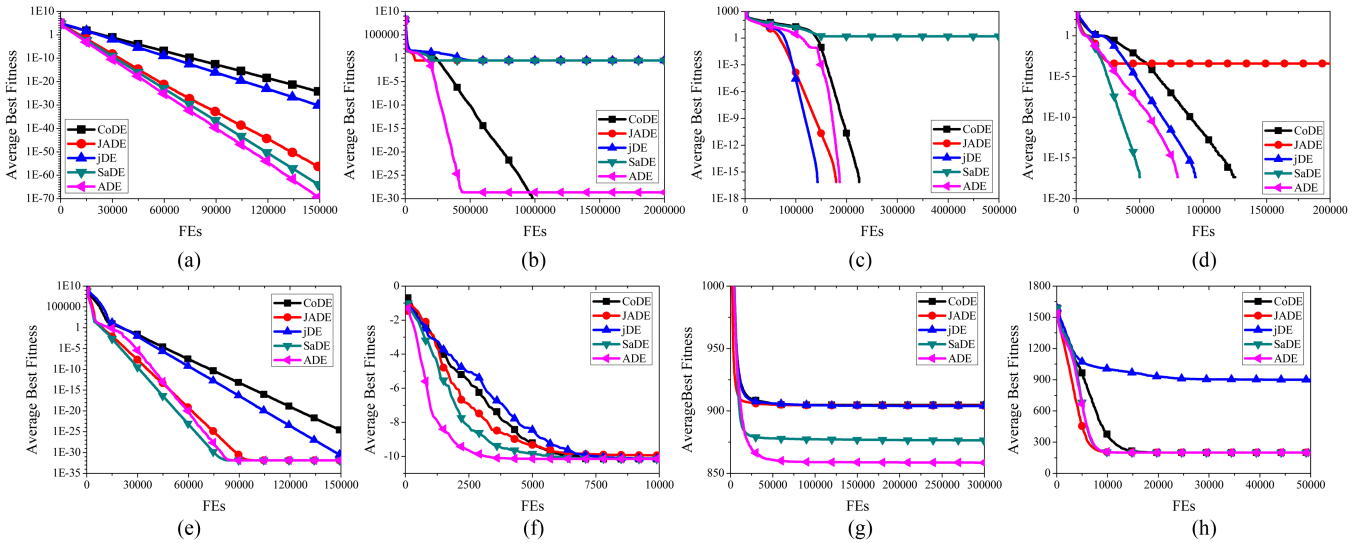


Fig. 3. Comparison of convergence performance between ADE and four state-of-the-art DE variants on eight benchmark functions. (a) f_1. (b) f_4. (c) f_8. (d) f_{10}. (e) f_{12}. (f) f_{20}. (g) f_{26}. (h) f_{32}.

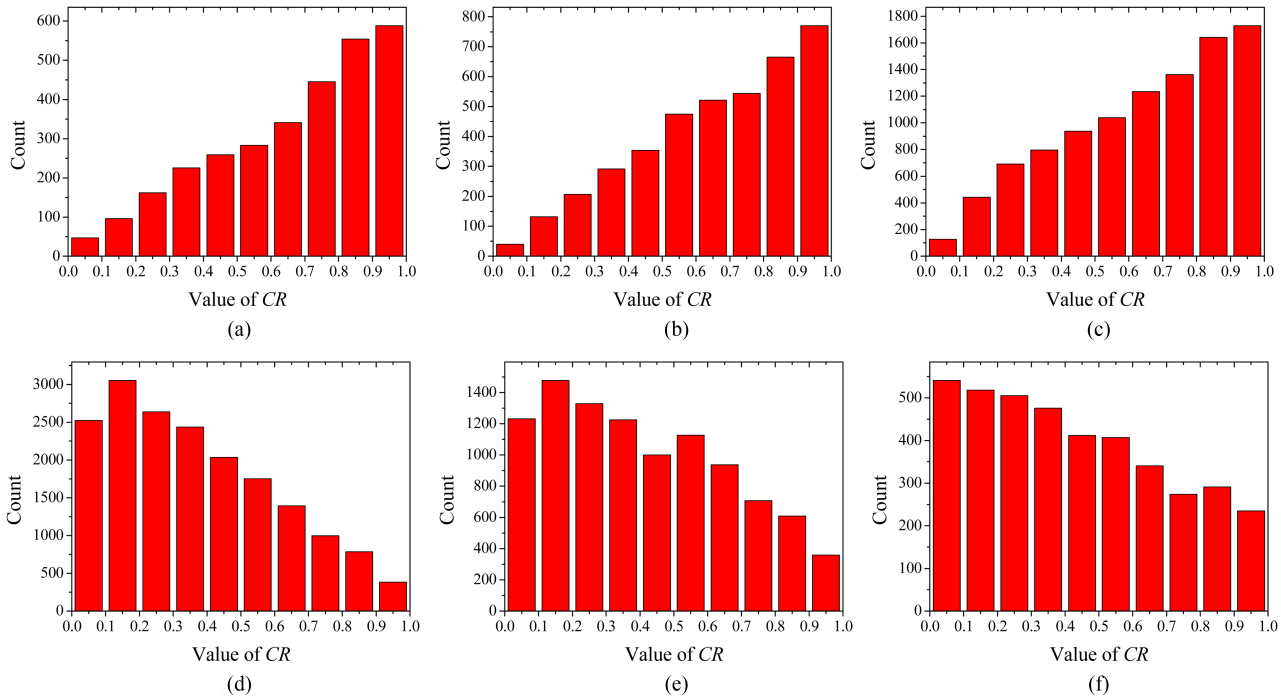


Fig. 4. Counts of different *CR* values during the evolutionary process of ADE on six benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_7. (e) f_8. (f) f_{10}.

on most of the multimodal functions, but it performs the worst on the unimodal functions. In contrast, DE/lbest/1 ($CR = 0.9$) obtains the most accurate results on the unimodal functions f_1–f_3, but it suffers from frequent premature convergence on all the multimodal functions. The performance of ADE-ponly is less dependent on the optimization problems. Not only can it get relatively high accuracy solutions on the unimodal functions, but also it is able to preserve premature convergence on most multimodal functions. These results demonstrate that the population-level parameter control strategy is helpful for improving robustness of the algorithm.

The effect of individual-level parameter control can be verified by a comparison between ADE and ADE-ponly. The results can be found in the first two columns of Table XII. One can observe that the ADE achieves solutions with slightly higher accuracy than ADE-ponly on the unimodal functions. Moreover, ADE also has a better global search ability to escape from local optima on the multimodal functions f_8 and f_9. The better performance of ADE should be due to the diversified parameter values brought by the individual-level parameter control.

E. Properties of F and CR of ADE

From the experimental results in Table XII, we observe that DE/lbest/1 performs better on f_1–f_3 when it is assigned a smaller value of *CR*, while a larger value of *CR* is more

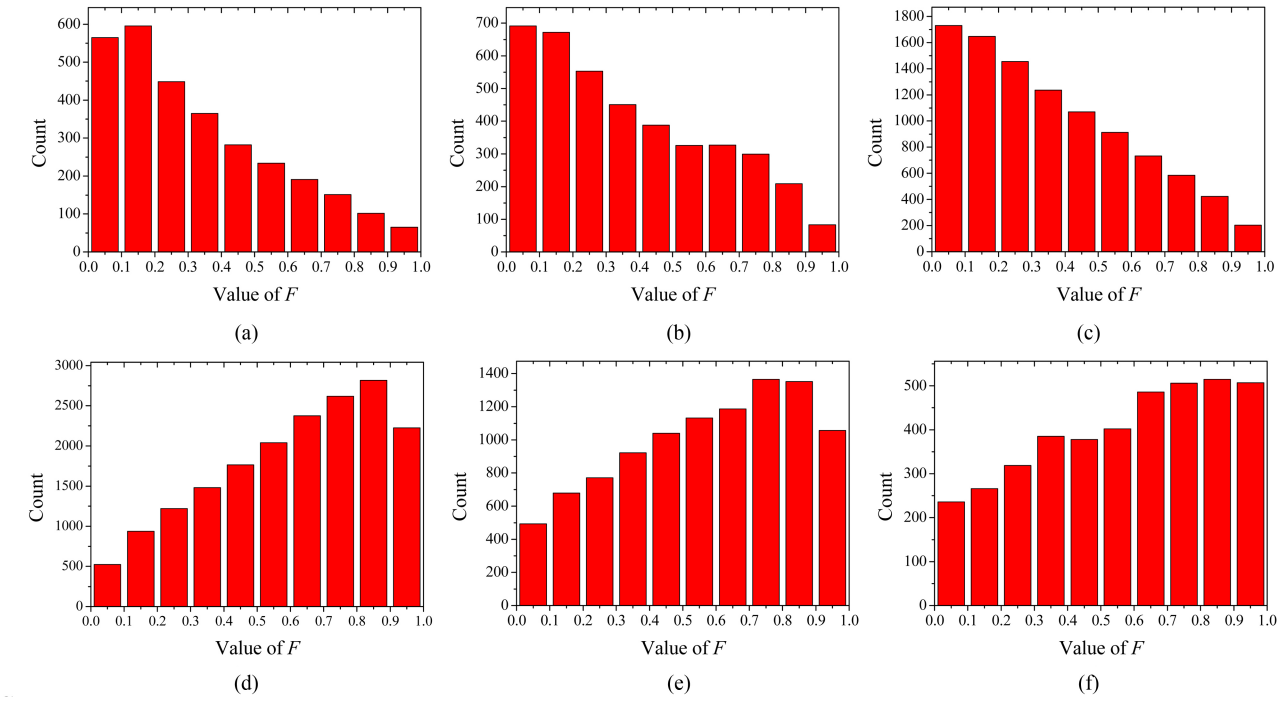


Fig. 5. Counts of different F values during the evolutionary process of ADE on six benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_7. (e) f_8. (f) f_{10}.

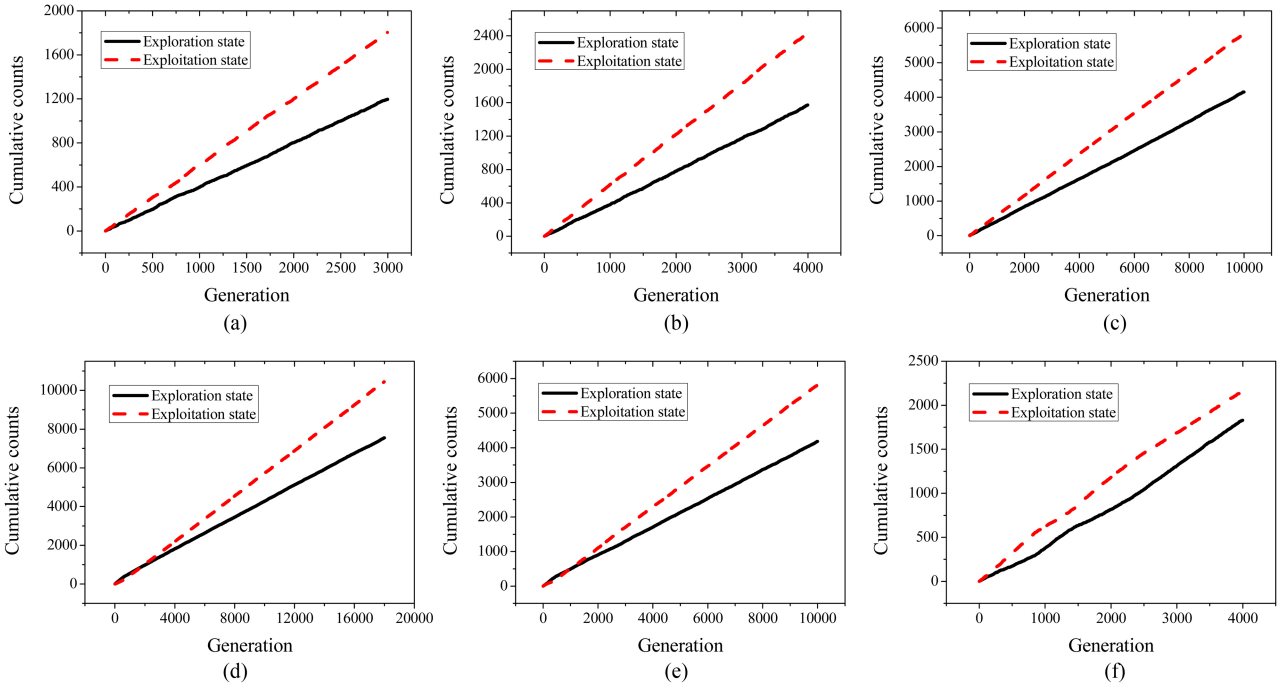


Fig. 6. Cumulative counts of exploration and exploitation states derived from DE/lbest/1 versus the number of generations on six benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_7. (e) f_8. (f) f_9.

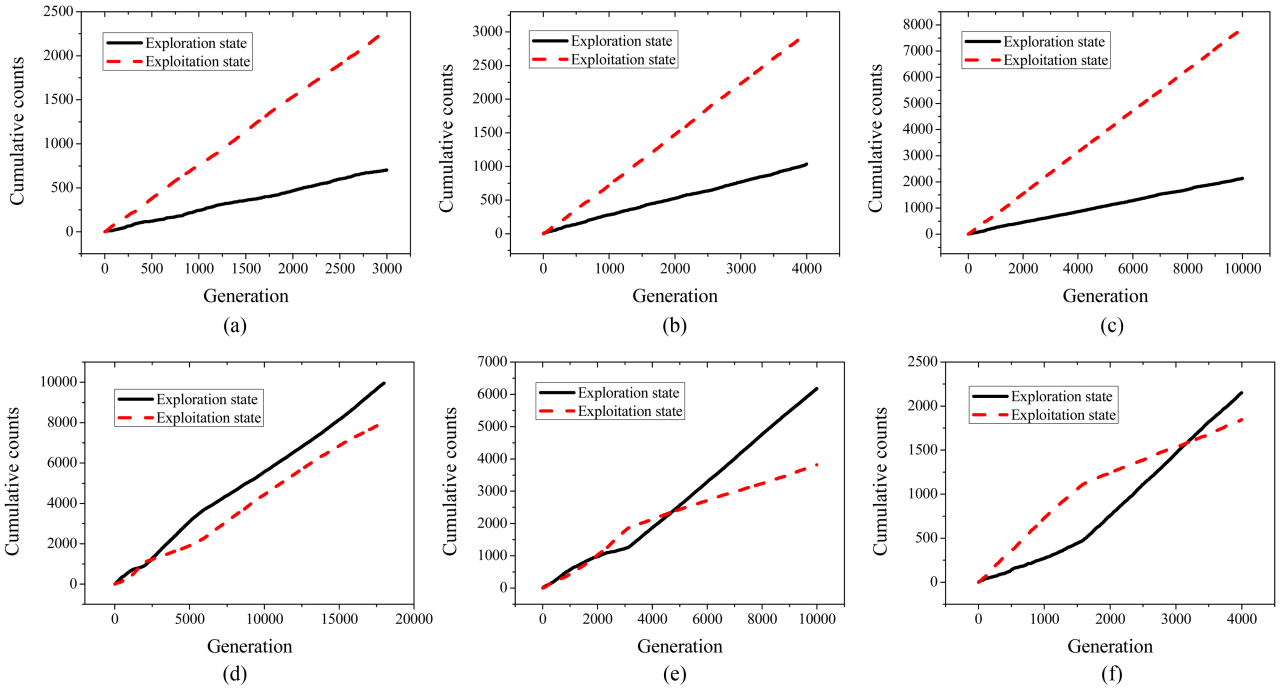


Fig. 7. Cumulative counts of exploration and exploitation states derived from ADE versus the number of generations on six benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_7. (e) f_8. (f) f_9.

beneficial to f_7, f_8, and f_{10}. Therefore, we can further analyze the property of CR value in ADE on these six functions to further check the effectiveness of parameter adaptation. We divide the range of population-level parameter CR_p [0,1] into ten intervals with equal lengths. In each generation, the value of CR_p is recorded. After a run of the ADE algorithm on a certain function, we compute the count of the CR_p value within each interval. Fig. 4 shows these counts on each test function. For the unimodal functions f_1–f_3, we find that the counts of CR_p keep increasing as the value of CR_p increases. On the contrary, for the multimodal functions f_7, f_8, and f_{10}, the counts of CR_p decrease in general with the value of CR_p, as expected.

In addition, we show the corresponding results of F_p in Fig. 5. It can be seen that F_p exhibits an opposite property of CR_p. Relatively small values of F_p are beneficial to finding solutions with high accuracy for unimodal functions. Conversely, relatively large values of F_p can help alleviating premature convergence for multimodal functions. Based on the above observation, it can be concluded that ADE can adapt the parameter effectively to meet the needs of different optimization problems.

F. Search Behavior of ADE

The search behavior of ADE is compared with that of DE/lbest/1 ($F = 0.5$, $CR = 0.9$) on three unimodal functions f_1–f_3 and three multimodal functions f_7–f_9. Figs. 6 and 7 illustrate the results of DE/lbest/1 and ADE, respectively, based on the data of a typical run. In the figures, the x-axis represents the number of generations and the y-axis represents the cumulative counts of optimization states (i.e., exploration and exploitation states) the algorithm has been in. The search behavior of algorithm can be analyzed by comparing these two cumulative counts during the evolution. According to Fig. 6, the cumulative count of the exploitation states is larger than that of the exploration states throughout the evolutionary process on both unimodal and multimodal functions. This is because DE/lbest/1 utilizes a greedy mutation strategy to some degree without parameter adaptation. The population is attracted by some locally best individuals, and thus the exploitation state always dominates the exploration state. Such a search behavior is good for unimodal functions, but it may lead to premature convergence for complex multimodal functions. According to Fig. 7, the search behavior of ADE on unimodal functions is similar to that of DE/lbest/1. The curve of the exploitation state is always beyond that of the exploration state. Furthermore, the gap between these two curves is larger than that of Fig. 6, which is beneficial to obtaining solutions with higher accuracy for unimodal functions. However, the search behavior of ADE is quite different from that of DE/lbest/1 on the three multimodal functions. The cumulative count of exploitation states is not always larger than that of the exploration states, but they dominate each other alternately during the evolutionary process. We take f_9 for example. The cumulative count of exploitation states increases faster than that of exploration states from the beginning to around 1500 generation. After that, this trend changes to the opposite, i.e., the exploration states begin to dominate the exploitation states. Finally, the cumulative count of exploration states overtakes that of exploitation states in around 3000 generation. For multimodal functions, it is important for the algorithm to keep exploring new regions of the search space and hence being trapped in local optima can be avoided. The ADE performs such a search behavior as we expected due to the parameter

TABLE X
SUCCESS RATE AND SEARCH SPEED COMPARISONS ON 100 *D* FUNCTIONS

Fun.	Acceptable accuracy		ADE	CoDE	JADE	jDE	SaDE
f_1	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	1.41E+05	1.96E+05	2.01E+05	5.76E+05	**1.02E+05**
		rank	2	3	4	5	**1**
f_2	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	2.66E+05	2.96E+05	3.08E+05	8.24E+05	**1.43E+05**
		rank	2	3	4	5	**1**
f_3	1E+02	SR	**100**	0	**100**	0	**100**
		FEs	2.74E+05	N/A	**1.83E+05**	N/A	2.81E+05
		rank	2	5	**1**	4	3
f_4	1E+02	SR	**100**	84	**100**	88	68
		FEs	4.68E+05	**4.14E+05**	5.13E+05	8.40E+05	6.43E+05
		rank	**1**	3	2	4	5
f_5	1E-10	SR	**100**	**100**	**100**	**100**	**100**
		FEs	5.40E+04	5.93E+04	**5.37E+04**	1.94E+05	2.03E+05
		rank	2	3	**1**	4	5
f_6	1E+00	SR	**100**	**100**	**100**	**100**	**100**
		FEs	1.96E+04	2.33E+04	2.06E+04	9.64E+04	**6.73E+03**
		rank	2	4	3	5	**1**
f_7	-40000	SR	**100**	**100**	**100**	**100**	28
		FEs	3.79E+05	3.65E+05	6.97E+05	7.58E+05	**2.40E+05**
		rank	2	**1**	3	4	5
f_8	1E+02	SR	**100**	**100**	**100**	**100**	**100**
		FEs	2.72E+05	5.54E+05	5.04E+05	4.67E+05	**2.00E+05**
		rank	2	5	4	3	**1**
f_9	1E+01	SR	**100**	**100**	**100**	**100**	**100**
		FEs	7.80E+03	1.62E+04	1.74E+04	9.88E+04	**4.71E+03**
		rank	2	3	4	5	**1**
f_{10}	1E+00	SR	**100**	**100**	**100**	**100**	**100**
		FEs	4.02E+04	5.67E+04	6.48E+04	1.83E+05	**2.55E+04**
		rank	2	3	4	5	**1**
f_{11}	1E+00	SR	**100**	**100**	**100**	**100**	**100**
		FEs	2.58E+05	5.63E+04	2.83E+05	2.01E+05	**3.00E+04**
		rank	4	2	5	3	**1**
f_{12}	1E+00	SR	**100**	**100**	**100**	**100**	80
		FEs	2.28E+05	**6.74E+04**	1.68E+05	2.30E+05	1.59E+05
		rank	3	**1**	2	4	5
avg_rank			**2.2**	3.0	3.1	4.3	2.5

TABLE XI
EFFECT OF DE/LEBST/1 MUTATION STRATEGY

Fun.	FEs	DE/lbest/1 (CR = 0.5) Mean (Std Dev)	DE/best/1 (CR = 0.5) Mean (Std Dev)	DE/current-to-$p$best/1 Mean (Std Dev)
f_1	150000	8.04E-70 (5.08E-31)	**6.17E-173 (0.00E+00)**	1.33E-67 (6.25E-68)
f_2	200000	5.01E-47 (4.58E-47)	**1.22E-122 (4.88E-122)**	1.78E-44 (5.22E-45)
f_3	500000	1.19E-03 (9.23E-04)	**2.55E-35 (7.12E-35)**	2.92E-15 (2.89E-15)
f_4	2000000	**4.78E-01 (1.30E+00)**	7.97E-01 (1.59E+00)	1.78E+01 (1.68E+00)
f_5	150000	**0.00E+00 (0.00E+00)**	3.92E+00 (5.99E+00)	**0.00E+00 (0.00E+00)**
f_6	300000	1.75E-03 (5.80E-04)	2.85E-03 (1.69E-03)	**8.82E-04 (2.31E-04)**
f_7	900000	-11599.82 (4.83E+02)	-10600.99 (4.89E+02)	**-11958.34 (2.21E+02)**
f_8	500000	**2.44E+01 (5.91E+00)**	3.18E+02 (9.83E+00)	6.17E+01 (5.24E+00)
f_9	200000	**4.14E-15 (0.00E+00)**	1.10E+00 (8.98E-01)	**4.14E-15 (0.00E+00)**
f_{10}	200000	**0.00E+00 (0.00E+00)**	2.77E-02 (5.89E-02)	2.96E-04 (1.48E-03)
f_{11}	150000	**1.09E-21 (5.33E-21)**	5.05E-01 (1.38E+00)	4.15E-03 (2.07E-02)
f_{12}	150000	**1.35E-32 (5.47E-48)**	2.94E-01 (7.68E-01)	**1.35E-32 (5.47E-48)**

TABLE XII
EFFECT OF TWO-LEVEL PARAMETER ADAPTATION

Fun.	FEs	ADE Mean (Std Dev)	ADE-ponly Mean (Std Dev)	DE/lbest/1 ($CR = 0.1$) Mean (Std Dev)	DE/lbest/1 ($CR = 0.5$) Mean (Std Dev)	DE/lbest/1 ($CR = 0.9$) Mean (Std Dev)
f_1	150000	1.49E-70 (3.95E-70)	1.46E-65 (3.60E-65)	5.83E-42 (3.62E-42)	8.04E-70 (5.08E-31)	**1.21E-74 (3.21E-74)**
f_2	200000	3.21E-51 (6.57E-51)	1.34E-49 (6.12E-49)	3.03E-31 (1.15E-31)	5.01E-47 (4.58E-47)	**8.66E-61 (2.63E-60)**
f_3	500000	8.13E-27 (2.60E-26)	5.05E-28 (2.23E-27)	4.53E+01 (1.55E+01)	1.19E-03 (9.23E-04)	**3.71E-57 (1.57E-56)**
f_4	2000000	**2.28E-29 (3.84E-29)**	6.38E-01 (1.46E+00)	1.76E+01 (5.45E+00)	4.78E-01 (1.30E+00)	1.75E+00 (1.98E+00)
f_5	150000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	2.00E+00 (2.73E+00)
f_6	300000	2.88E-03 (1.17E-03)	3.09E-03 (1.49E-03)	4.89E-03 (1.01E-03)	**1.75E-03 (5.80E-04)**	6.51E-03 (2.59E–03)
f_7	900000	**-12569.49 (1.82E-12)**	**-12569.49 (1.82E-12)**	**-12569.49 (1.82E-12)**	-11599.82 (4.83E+02)	-6942.89 (3.89E+02)
f_8	500000	**0.00E+00 (0.00E+00)**	7.16E-01 (9.13E-01)	7.96E-01 (7.45E-01)	2.44E+01 (5.91E+00)	9.01E+01 (1.16E+01)
f_9	200000	5.99E-15 (1.77E-15)	5.06E-01 (6.50E-01)	8.69E-15 (1.60E-15)	**4.14E-15 (0.00E+00)**	1.40E+00 (6.95E-01)
f_{10}	200000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	7.88E-03 (1.93E-02)
f_{11}	150000	**1.57E-32 (2.74E-48)**	1.74E-32 (5.21E-33)	1.64E-32 (3.79E-33)	1.09E-21 (5.33E-21)	8.07E-01 (1.38E+00)
f_{12}	150000	**1.35E-32 (5.47E-48)**	1.38E-32 (1.02E-33)	**1.35E-32 (5.47E-48)**	**1.35E-32 (5.47E-48)**	6.61E-02 (3.13E-01)

adaptation, which adjusts the parameters to suitable values that allow the algorithm to perform more explorative behavior.

G. Sensitivities to Population Parameters

As mentioned in Section III-A, the mutation strategy of ADE is designed to achieve a better balance between exploration and exploitation. The best vector involved in mutation is selected from its respective group of vectors. Thus, the performance of ADE may be sensitive to the selected population size and the number of groups the population divided into. If the population size is large and the number of groups is small, which means the best vector is selected from a large group of individuals, the mutation strategy would be relatively greedy. On the contrary, if the best vector is selected from a small group of individuals and the number of groups is large, then the mutation strategy becomes relatively diverse. In order to investigate the sensitivity of ADE to the combination of the population size and the number of groups, we compare ADE with different values of these two parameters. The ADE algorithm with a population size of 50 and ten groups, for example, is denoted as ADE-p50g10. Table XIII reports the experimental results on f_1–f_{12} with $D = 30$. In general, for the simple unimodal functions such as f_1–f_3, using a smaller population size and a smaller number of groups is more helpful to get solutions with high accuracy. However, such a setting suffers from frequent premature convergence on multimodal functions. On the contrary, for multimodal functions, using a larger population size and a larger number of groups is better for maintaining diversity and preventing premature convergence. In addition, it can be observed that when the number of groups is set to 10, the performance of ADE is not so sensitive to the population size. Overall, a population size of 50 and dividing the population into ten groups is a suitable combination to balance the performance on both unimodal and multimodal functions.

H. Results on Real-World Optimization Problems

The performance of the ADE is further tested on three real-world optimization problems from the IEEE CEC 2011 competition [46]. The three test problems are parameter estimation for frequency-modulated sound waves (P1), Lennard–Jones potential problem (P2), and bifunctional catalyst blend optimal control problem (P3). More details of these problems can be found in [46].

Table XIV compares the mean and standard deviation of the final results over 25 runs of the five DE variants. The algorithms are executed for 50 000, 100 000, and 150 000 FEs, respectively, on all the problems. From Table XIV, it can be noted that ADE remains a competitive approach among the five DE variants. In the case of problem P1, ADE significantly outperforms the other four DE variants. JADE performs the best on problem P2 and ADE is the second best. For problem P3, all the five DE variants present similar results without significant difference. These observations demonstrate that the proposed ADE has some potential for solving real-world optimization problems.

I. Discussions

To improve the efficiency and robustness of DE, we propose a greedy mutation strategy DE/lbest/1 and a two-level parameter adaptation scheme for DE. Experiments are extensively conducted on 33 benchmark functions and three real-world optimization problems. From the experimental results, the strengths and weaknesses of the proposed ADE can be summarized as follows.

1) In the case of unimodal functions, the overall performance of ADE is similar to that of JADE and SaDE, but better than CoDE and jDE. However, ADE is beaten by the classic DE/best/1 on three simple unimodal functions due to the high greediness degree of the latter. Importantly, only ADE can achieve 100% success rates on all these functions.
2) In the cases of multimodal functions and hybrid composition functions, ADE, JADE, and SaDE converge faster than CoDE and jDE in general. CoDE and jDE are more robust than JADE and SaDE in some cases since the formers utilize more diverse mutation strategies. Note that ADE still shows the most reliable performance in all cases in spite of its greedy DE/lbest/1 mutation strategy. Also, ADE achieves the best final accuracy results on most of these functions.
3) The ranks of ADE in the success rates and search speed comparison tables may be close to that of its competitor in some instances. For example, the average ranks of ADE and JADE are 2 and 2.1, respectively, in Table VIII. It is found that the small average rank of JADE largely stems from its fast search speed on most

TABLE XIII
ADE WITH DIFFERENT POPULATION PARAMETERS

Fun.	FEs	ADE-p50g10 Mean (Std Dev)	ADE-p100g10 Mean (Std Dev)	ADE-p50g5 Mean (Std Dev)	ADE-p100g5 Mean (Std Dev)
f_1	150000	1.49E-70 (3.95E-70)	1.40E-55 (2.68E-55)	**6.66E-99 (1.63E–98)**	1.01E-80 (3.80E-80)
f_2	200000	3.21E-51 (6.57E-51)	4.68E-39 (5.65E-39)	**8.61E–80 (1.61E–79)**	4.78E–60 (1.45E–59)
f_3	500000	8.13E-27 (2.60E-26)	3.04E-22 (4.99E-22)	**5.20E–37 (1.74E–36)**	6.93E-31 (1.72E-30)
f_4	2000000	**2.28E-29 (3.84E-29)**	1.59E-01 (7.81E-01)	6.38E-01 (1.46E+00)	1.12E+00 (1.79E+00)
f_5	150000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.60E+00 (1.96E+00)	1.28E+00 (3.94E+00)
f_6	300000	2.88E-03 (1.17E-03)	2.88E-03 (1.13E-03)	4.69E–03 (2.23E–03)	**2.86E–03 (9.38E–04)**
f_7	900000	**-12569.49 (1.82E-12)**	**-12569.49 (1.82E-12)**	-12440.78 (1.71E+02)	-12526.85 (8.12E+01)
f_8	500000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	2.55E+00 (1.59E+00)	1.95E+00 (1.90E+00)
f_9	200000	5.99E-15 (1.77E-15)	**4.71E-15 (1.30E-15)**	1.34E+00 (1.18E+00)	2.37E-01 (5.94E-01)
f_{10}	200000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.00E-02 (1.58E-02)	6.39E-03 (1.19E-02)
f_{11}	150000	**1.57E-32 (2.74E-48)**	**1.57E-32 (2.74E-48)**	3.32E-02 (1.24E-01)	3.74E-02 (1.64E-01)
f_{12}	150000	**1.35E-32 (5.47E-48)**	**1.35E-32 (5.47E-48)**	8.79E-04 (2.98E-03)	1.43E-32 (2.15E-33)

TABLE XIV
COMPARISON BETWEEN ADE AND FOUR STATE-OF-THE-ART DE VARIANTS ON THREE REAL-WORLD OPTIMIZATION PROBLEMS

Problem	FEs	ADE Mean (Std Dev)	CoDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)
P1	50000	**8.19E-02 (1.87E-01)**	1.18E+00† (3.25E+00)	3.56E+00† (3.53E+00)	1.10E+01† (3.79E+00)	9.57E+00† (5.32E+00)
	100000	**1.36E-03 (5.18E-03)**	9.46E-01† (3.28E+00)	7.02E-01† (9.36E-01)	3.09E+00† (2.56E+00)	6.49E-01† (3.73E+00)
	150000	**4.12E-04 (1.76E-03)**	8.13E-01† (2.82E+00)	5.11E-01† (1.71E+00)	2.60E-01† (6.38E–01)	4.92E-01† (2.46E+00)
P2	50000	-1.55E+01 (2.34E+00)	-1.16E+01† (1.12E+00)	**-1.86E+01† (1.06E+00)**	-1.33E+01† (1.89E+00)	-1.05E+01† (7.92E-01)
	100000	-1.90E+01 (2.50E+00)	-1.48E+01† (1.20E+00)	**-2.22E+01† (9.42E-01)**	-1.75E+01† (1.45E+00)	-1.29E+01† (1.14E+00)
	150000	-2.20E+01 (1.99E+00)	-1.85E+01† (2.79E+00)	**-2.39E+01† (7.25E-01)**	-1.98E+01† (1.08E+00)	-1.43E+01† (1.06E+00)
P3	50000	**1.15E-05 (1.26E-19)**	**1.15E-05 (2.38E-19)**	**1.15E-05 (1.64E-19)**	**1.15E-05 (2.17E-19)**	**1.15E-05 (1.46E-19)**
	100000	**1.15E-05 (1.46E-19)**	**1.15E-05 (2.49E-19)**	**1.15E-05 (1.84E-19)**	**1.15E-05 (1.34E-19)**	**1.15E-05 (2.70E-19)**
	150000	**1.15E-05 (1.24E-19)**	**1.15E-05 (1.45E-19)**	**1.15E-05 (1.47E-19)**	**1.15E-05 (1.32E-19)**	**1.15E-05 (2.38E-19)**

† THE DIFFERENCE BETWEEN THE RESULTS OF ADE AND THE CORRESPONDING ALGORITHM IS SIGNIFICANT AT $\alpha = 0.05$ BY WILCOXONS'S RANK SUM TEST

functions in this group. On the other hand, in the case of ADE both algorithm reliability (indicated by success rate) and search speed contribute to the small average rank. A close inspection of Table VIII indicates that JADE converges fastest for the optimization of functions f_{26}–f_{33}. ADE is beaten by JADE on these functions, but still converges second fastest. However, JADE shows the worst performance in terms of reliability on the other three functions f_{23}–f_{25}. On the contrary, ADE achieves the highest success rates on all test functions. Moreover, when the problem dimension increases to 100, the difference between the average rank of ADE and that of JADE becomes more obvious. The advantage of ADE is that it benefits from fast convergence without sacrificing its robustness.

4) Regarding the results of real-world problems, a close inspection of Table XIV reveals that ADE is the only one that may provide consistent good performance on all test problems. Although ADE is beaten by JADE on problem P2, it is still the second best among the five DE variants. Conversely, CoDE, jDE, and SaDE show relatively poor or fair performance on both problems P1 and P2. Even for JADE that is the best on problem P2, it performs the second worst on problem P1 in terms of final accuracy. Since the problem characteristics are usually not known *a priori* when solving real-world problems, it should be a good choice to use an algorithm such as ADE that is expected to perform favorably on problems with various characteristics.
5) Overall, by combining the DE/lbest/1 mutation strategy with the two-level parameter adaptation scheme, ADE is efficient in terms of solution quality and search speed, while maintaining the algorithm reliability at a high level.

V. CONCLUSION

In this paper, a new DE variant called ADE has been proposed. The ADE algorithm is characterized by a novel mutation strategy DE/lbest/1 and an adaptive parameter control strategy. The DE/lbest/1 strategy utilizes the information of several locally best solutions instead of the single globally best solution used in the classic DE/best/1. Such a mutation strategy is helpful for balancing the fast convergence and population diversity of the algorithm. Two levels of parameters, i.e., population-level and individual-level parameters, have been introduced into the parameter adaptation scheme of ADE. The population-level parameters are first controlled based on an optimization state estimation. Then, the individual-level parameters for each individual are generated by adjusting the population-level parameters according to the individual fitness rank and its distance rank from the best individual.

We have tested ADE on a suite of benchmark functions and three real-world optimization problems. The results show that ADE exhibits appropriate search behavior for both unimodal and multimodal problems. The performance of ADE is compared with four state-of-the-art DE variants, i.e., CoDE, JADE, jDE, and SaDE. It can be concluded that ADE performs better than, or at least comparably to, the other DE variants

in terms of solution quality, convergence speed, and algorithm reliability. However, ADE cannot perform the best for all kinds of optimization problems. For example, the DE/best/1 obtains solutions with higher accuracy than ADE on three simple unimodal problems due to the greediness of DE/best/1, which can be observed from Table XI. ADE only can achieve a balanced performance for different kinds of optimization problems. In addition, the effects of ADE components, parameter properties of ADE, and parameter sensitivity of ADE have been studied.

For future work, we will extend the ADE algorithm to solving other complex optimization problems such as multiobjective and dynamic optimization problems. In addition, we can regard the proposed two-level parameter adaptation strategy as a general two-level parameter control framework and apply it to other EAs.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor, the anonymous reviewers, and Prof. Y.-H. Shi for their valuable comments and suggestions on this paper.

REFERENCES

[1] R. M. Storn and K. V. Price. (1995). "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," TR-95-012 [Online]. Available: http://icsi.berkeley.edu/storn/litera.html

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

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

[4] 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.

[5] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Comput*., vol. 15, no. 11, pp. 2175–2185, 2011.

[6] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern*., vol. 43, no. 3, pp. 881–897, Jun. 2013.

[7] Y. Wang and Z. X. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Trans. Evol. Comput*., vol. 16, no. 1, pp. 117–134, Feb. 2012.

[8] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput*., vol. 16, no. 5, pp. 601–614, Oct. 2012.

[9] A. Basak, S. Das, and K. C. Tan, "A bi-objective differential evolution algorithm enhanced with mean distance based selection for multimodal optimization," *IEEE Trans. Evol. Comput*., to be published.

[10] R. Storn, "Differential evolution design of an IIR-filter with requirements for magnitude and group delay," in *Proc. IEEE Int. Conf. Evol. Comput*., 1996, pp. 268–273.

[11] R. Joshi and A. C. Sanderson, "Minimal representation multi-sensor fusion using differential evolution," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 1, pp. 63–76, Jan. 1999.

[12] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "An evolutionary multi-objective sleep scheduling scheme for differentiated coverage in wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1093–1102, Nov. 2012.

[13] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Trans. Syst., Man, Cybern. B, Cybern*., vol. 42, no. 1, pp. 107–124, Feb. 2012.

[14] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Commun*., vol. 22, no. 1, pp. 1–20, Jan. 2009.

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

[16] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proc. IEEE Congr. Evol. Comput*., 2005, pp. 506–513.

[17] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and nonseparability: Using selective pressure to focus search," in *Proc. Genet. Evol. Comput. Conf*., Jul. 2007, pp. 1428–1435.

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

[19] M. M. Ali and A. Törn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Comput. Oper. Res*., vol. 31, no. 10, pp. 1703–1725, 2004.

[20] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput. A Fusion Found. Methodol. Appl.*, vol. 9, no. 6, pp. 448–462, Apr. 2005.

[21] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proc. 9th Int. Conf. MENDEL*, Jun. 2003, pp. 41–46.

[22] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proc. 5th Int. Conf. Parallel Process. Appl. Math.*, Sep. 2003, pp. 261–268.

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

[24] 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.

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

[26] 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.

[27] 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.

[28] Y. Wang, Z. X. Cai, and Q. F. 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.

[29] Y. Zhong and L. Zhang, "Remote sensing image subpixel mapping based on adaptive differential evolution," *IEEE Trans. Syst., Man, Cybern. B, Cybern*., vol. 42, no. 5, pp. 1306–1329, Oct. 2012.

[30] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Inf. Sci*., vol. 181, no. 18, pp. 3749–3765, 2011.

[31] W.-J. Yu and J. Zhang, "Adaptive differential evolution with optimization state estimation," in *Proc. Genet. Evol. Comput. Conf*., Jul. 2012, pp. 1285–1292.

[32] K. V. Price, "An introduction to differential evolution," in *New Ideas in Optimization*. London, U.K.: McGraw-Hill, 1999, pp. 79–108.

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

[34] P. Kaelo and M. M. Ali, "Differential evolution algorithms using hybrid mutation," *Comput. Optimization Appl*., vol. 37, pp. 231–246, Jun. 2007.

[35] 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.

[36] 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.

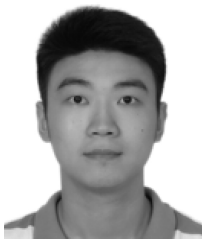
[37] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern*., vol. 42, no. 2, pp. 397–413, Apr. 2012.

[38] 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.

[39] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern*., to be published.

[40] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern*., to be published.

[41] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genet. Evol. Comput. Conf*., 2006, pp. 485–492.
[42] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput*., vol. 3, no. 2, pp. 82–102, Jul. 1999.
[43] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definition and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, IIT Kanpur, Kanpur, India, Tech. Rep. KanGAL#2005005, May 2005.
[44] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[45] Y.-W. Shang and Y.-H. Qiu, "A note on the extended Rosenbrock function," *Evol. Comput*., vol. 14, no. 1, pp. 119–126, 2006.
[46] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Dept. Electron. Telecommun. Engg., Jadavpur Univ., Kolkata, India, Nanyang Technol. Univ., Singapore, Tech. Rep., 2010.



Wei-Jie Yu (S'10) received the bachelor's degree in network engineering from Sun Yat-Sen University, Guangzhou, China, in 2009, where he is currently pursuing the Ph.D. degree.

His current research interests include differential evolution, artificial bee colony optimization, ant colony optimization, particle swarm optimization, and their applications to real-world problems.

Meie Shen received the B.S. degree in industrial automation from the Huazhong University of Science and Technology, Wuhan, China, in 1986, and the M.S. degree in automatic control from the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 1989.

She is currently an Associate Professor with the School of Computer Science, Beijing Information Science and Technology University, Beijing, China. Her current research interests include intelligent algorithms, automatic control theory, and applications.

Wei-Neng Chen (SM'07–M'12) received the bachelor's degree in network engineering and the Ph.D. degree in computer application technology from Sun Yat-Sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Lecturer with the Department of Computer Science, Sun Yat-Sen University. His current research interests include swarm intelligence algorithms and their applications to cloud computing, financial optimization, operations research and software engineering.

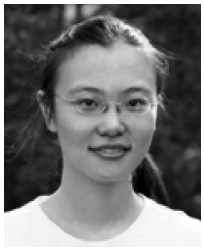
Dr. Chen's doctoral dissertation was awarded the China Computer Federation Outstanding Dissertation in 2012.

Zhi-Hui Zhan (M'13) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-Sen University, Guangzhou, China, in 2007 and 2013, respectively.

His current research interests include particle swarm optimization, differential evolution, ant colony optimization, genetic algorithms, brain storm optimization, and their applications to real-world problems.

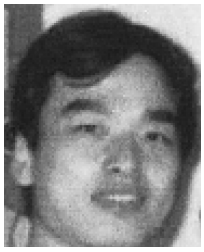
Yue-Jiao Gong (S'10) received the B.S. degree in computer science from Sun Yat-Sen University, Guangzhou, China, in 2010, where she is currently pursuing the Ph.D. degree.

Her current research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications to design and optimization of intelligent transportation systems, hospital management systems, and RFID systems.



Ying Lin (M'12) received the Ph.D. degree in computer applied technology from Sun Yat-Sen University, Guangzhou, China, in 2012,

She is currently a Lecturer with the Department of Psychology, Sun Yat-Sen University. Her current research interests include computational intelligence and its applications to scheduling, sentiment tracking, and cognitive diagnostic modeling.



Ou Liu received the Ph.D. degree in information systems from the City University of Hong Kong, Kowloon, Hong Kong, in 2006.

Since 2006, he has been with Hong Kong Polytechnic University, Hung Hom, Hong Kong, where he is currently an Assistant Professor with the School of Accounting and Finance. His current research interests include business intelligence, decision support systems, ontology engineering, genetic algorithms, ant colony system, fuzzy logic, and neural networks.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

From 2003 to 2004, he was a Brain Korean 21 Post-Doctoral Fellow with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. Since 2004, he has been with Sun Yat-Sen University, Guangzhou, China, where he is currently a Cheung Kong Professor with the Department of Computer Science. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. His current research interests include computational intelligence, cloud computing, high-performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON CYBERNETICS, and the *IEEE Computational Intelligence Magazine*. He is the Founding and Current Chair of the IEEE Guangzhou Subsection and IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters.