Adaptive Distributed Differential Evolution

Zhi-Hui Zhan, *Senior Member, IEEE*, Zi-Jia Wang, *Student Member, IEEE*, Hu Jin, *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Due to the increasing complexity of optimization problems, distributed differential evolution (DDE) has become a promising approach for global optimization. However, similar to the centralized algorithms, DDE also faces the difficulty of strategies' selection and parameters' setting. To deal with such problems effectively, this article proposes an adaptive DDE (ADDE) to relieve the sensitivity of strategies and parameters. In ADDE, three populations called exploration population, exploitation population, and balance population are co-evolved concurrently by using the master–slave multipopulation distributed framework. Different populations will adaptively choose their suitable mutation strategies based on the evolutionary state estimation to make full use of the feedback information from both individuals and the whole corresponding population. Besides, the historical successful experience and best solution improvement are collected and used to adaptively update the individual parameters (amplification factor *F* and crossover rate CR) and population parameter (population size *N*), respectively. The performance of ADDE is evaluated on all 30 widely used benchmark functions from the CEC 2014 test suite and all 22 widely used real-world application problems from the CEC 2011 test suite. The experimental results show that ADDE has great superiority compared with the other state-of-the-art DDE and adaptive differential evolution variants.**

***Index Terms*—Adaptive distributed differential evolution (ADDE), differential evolution (DE), evolutionary state estimation (ESE), historical successful experience (HSE), master–slave multipopulation distributed.**

I. Introduction

DIFFERENTIAL evolution (DE), proposed by Storn and Price [1], is a kind of evolutionary algorithm (EA) that solves the optimization problems by the iterations of evolutionary operators, including mutation, crossover, and selection [2]–[7]. These days, DE has been successfully applied to a wide range of optimization problems, such as railway timetable scheduling [8]; biology research [9], [10]; and multivalued logic networks [11].

However, with the rapid development of information science and technology, the increasing complexity of the problems has posed new challenges to DE since the search space involves a huge number of local optima [12]–[17]. The popularization of distributed computing, which deploys the population on distributed systems, provides a novel way to improve the availability and realize more powerful performance [18]–[20]. Some efforts have been paid to use the distributed computing to enhance the performance of DE. For example, in parallel DE (PDE) [21], the best individual in each parallel subpopulation migrates to the next subpopulation in the ring topology. In Island-based distributed DE (IBDDE) [22], two subpopulations are co-evolved using random immigrant mechanism. Cloud-ready DE (Cloudde) [23], proposed by Zhan *et al.*, keeps four parallel subpopulations with different fixed mutation strategies and parameters, where the subpopulation with better performance will attract other individuals from poorly performed subpopulations. Ge *et al.* [24] proposed a distributed DE (DDE) with adaptive mergence and split strategy (DDE-AMS) for large-scale optimization, where the number and the size of subpopulations adaptively change for better performance. In asynchronous adaptive multipopulation DDE (AsAMP-dDE) [25], the multipopulation is combined with the asynchronous migration mechanism for exchanging information.

These DDE variants have shown their promising effectiveness on some complex optimization problems. However, similar to the centralized algorithms, DDEs also face the difficulty of strategies' selection and parameters' setting. That is, the performance of DDEs still greatly depends on the strategies' selection [26]–[31] and parameters' setting, including the amplification factor *F*, crossover rate CR, and population size *N* [32]–[34]. Different mutation strategies and parameters are suitable for different individuals and different populations. Therefore, the mutation strategies and the parameters need to be adaptively controlled to meet the searching requirement of each individual/population. Most of the

Manuscript received June 18, 2019; revised September 3, 2019; accepted September 21, 2019. Date of publication October 21, 2019; date of current version October 26, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, in part by the Guangdong–Hong Kong Joint Innovation Platform under Grant 2018B050502006, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) under Grant NRF-2019H1D3A2A01101977. This article was recommended by Associate Editor H. Takagi. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Z.-H. Zhan is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

Z.-J. Wang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

H. Jin and J. Zhang are with Hanyang University, Ansan 15588, South Korea (e-mail: junzhang@ieee.org).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

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

Digital Object Identifier 10.1109/TCYB.2019.2944873

current studies mentioned above [21]–[25], only use the fixed parameter and mutation strategy, and cannot achieve the adaptive control.

Therefore, this article proposes an adaptive DDE (ADDE) to address this issue. The ADDE algorithm uses a master–slave multipopulation distributed framework, cooperatively with both adaptive mutation strategy selection and adaptive parameters' settings based on the evolutionary state estimation (ESE), historical successful experience (HSE), and best solution improvement (BSI). Specifically, ADDE is promising by using the following three major novel advantages.

1) Three populations called exploration population, exploitation population, and balance population co-evolve concurrently by using the master–slave multipopulation distributed framework. Significantly, the composition of these three populations is dynamically changed according to the iteration process which can maximize their strengths by cooperation and enhance the global optimality of DDE.
2) Different populations will adaptively choose their suitable mutation strategies based on the ESE, which can make full use of the feedback information from both individuals and the whole corresponding population.
3) The HSE and BSI are collected and used to adaptively update the individual parameters (F and CR) and population parameter (N), respectively.

As we can see, ADDE does not solely adaptively control the mutation strategy selection or parameter setting, but adaptively controls these two components. The experimental results on the CEC 2014 benchmark test suite and the CEC 2011 real-world application problems, compared with the other state-of-the-art DDE and adaptive DE (ADE) variants, even the winner of the CEC 2014, have fully illustrated the effectiveness of ADDE.

The remainder of this article is organized as follows. Section II describes the basic DE and its developments. Section III describes the proposed ADDE algorithm in detail. In Section IV, the experimental results are presented and discussed. Finally, Section V draws the conclusions.

II. DE, DDE, AND ADE

A. DE

DE is a population-based searching method which evolves according to the difference between individuals. The initial population is randomly generated according to a uniform distribution in the search space as

$$x_{i,j,0} = L_j + \text{rand} \times \left(U_j - L_j\right) \tag{1}$$

where i is the individual index; L_j and U_j are the predefined lower and upper bounds of the jth dimension; and *rand* is a random number in range [0, 1]. After initialization, DE will evolve by a loop of evolutionary operators, including mutation, crossover, and selection until reaching the terminal condition. The operators are described as below.

Mutation: At each generation g, each individual $\boldsymbol{x}_{i,g}$ will generate its corresponding mutant vector $\boldsymbol{v}_{i,g}$ by the mutation operator as

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{r1,g} + F \times \left(\boldsymbol{x}_{r2,g} - \boldsymbol{x}_{r3,g}\right) \tag{2}$$

where $r1$, $r2$, and $r3$ are the three distinct random integers selected from $\{1, 2, \ldots, N\}$, which are all different from index i. The parameter F is a positive control parameter called the amplification factor, which controls the amplification of the difference vector.

This is called DE/rand/1 mutation strategy because the mutation used a random $\boldsymbol{x}_{r1,g}$ as the base and is disturbed by 1 difference provided by $(\boldsymbol{x}_{r2,g} - \boldsymbol{x}_{r3,g})$. Some other popular mutation strategies are also proposed in the literature such as the DE/best/1 mutation strategy as

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{\text{best},g} + F \times \left(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}\right) \tag{3}$$

where $\boldsymbol{x}_{\text{best},g}$ is the individual with the best fitness in the gth generation.

It should be noted that for each dimension j, some components of the mutant vector may exceed the predefined boundary constraints, which means $v_{i,j,g}$ is infeasible, it is reset by the boundaries as

$$v_{i,j,g} = \begin{cases} L_j, & \text{if } v_{i,j,g} < L_j \\ U_j, & \text{if} v_{i,j,g} > U_j. \end{cases} \tag{4}$$

Crossover: After mutation, DE generally performs a binomial crossover operator between $\boldsymbol{x}_{i,g}$ and $\boldsymbol{v}_{i,g}$ to generate a trial vector $\boldsymbol{u}_{i,g}$

$$u_{i,j,g} = \begin{cases} v_{i,j,g} & \text{if rand} \le \text{CR or } j = j_{\text{rand}} \\ x_{i,j,g} & \text{otherwise} \end{cases} \tag{5}$$

where j_{rand} is a random integer selected from $\{1, 2, \ldots, D\}$, used to ensure that $\boldsymbol{u}_{i,g}$ has at least one dimension comes from $\boldsymbol{v}_{i,g}$. The crossover rate CR is an another control parameter, which determines the fraction of $\boldsymbol{u}_{i,g}$ inherited from $\boldsymbol{v}_{i,g}$.

Selection: To determine whether the new generated individual $\boldsymbol{u}_{i,g}$ will survive in the next generation, the $\boldsymbol{u}_{i,g}$ is compared with the $\boldsymbol{x}_{i,g}$. The vector with better fitness enters the next generation. For example, for a minimization problem, the vector with lower fitness value enters the next generation, shown as

$$\boldsymbol{x}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g}, & \text{if } f\left(\text{u}_{i,g}\right) \le f\left(\text{x}_{i,g}\right) \\ \text{x}_{i,g}, & \text{otherwise} \end{cases} \tag{6}$$

where $f(\boldsymbol{x})$ is the fitness evaluation function.

B. Developments of DDE

With the huge development of computational resources, distributed computing has emerged as a form of high-performance computation, where many calculations are carried out concurrently. In [35], the master–slave model is incorporated with DE, where each member of the population is evaluated independently. In [21]–[25], the entire population is divided into different subpopulations, and DE is executed within each subpopulation concurrently toward the optimal solution. Specifically, in PDE [21], it maintains several parallel subpopulations and the best individual in each parallel subpopulation migrates to the next subpopulation in the ring topology. Such a ring topology has also been utilized in [36] and [37]. In IBDDE [22], two subpopulations are co-evolved using random

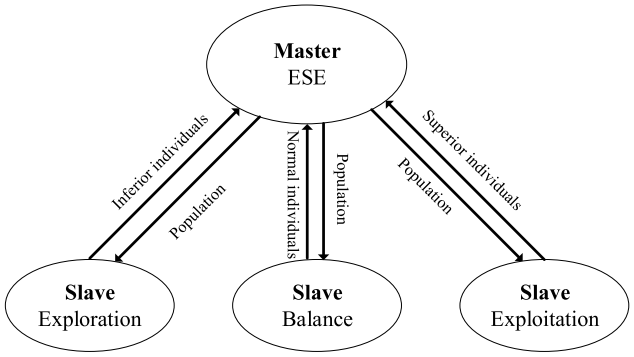


Fig. 1. Master–slave multipopulation distributed model.

immigrant mechanism. Cloudde [23], proposed by Zhan *et al*, keeps four parallel subpopulations with different fixed mutation strategies and parameters, where the subpopulation with better performance will attract other individuals from poorly performed subpopulations. Ge *et al.* [24] proposed the DDE-AMS for large-scale optimization, where the number and the size of subpopulations are adaptively changed for better performance. Meanwhile, they also proposed the DDE with space-driven topology (DDE-SD) [38], where the topology is constructed and updated according to the distances and contributions among different subpopulations. In AsAMP-dDE [25], the multipopulation is combined with the asynchronous migration mechanism for exchanging information. In DDE with multicultural migration (DDEM) [39], which makes use of two migration selection approaches, maintains a high diversity in the subpopulations. It shows DDEM with center individual-based migrant selection (RIBMS/C) and affinity-based replacement selection (ABRS) can obtain the best results, and such a DDEM variant is called DDEM-RCA.

For applications, Xu *et al.* [40] and Glotić *et al.* [41] applied DDE into the power systems, De Falco *et al.* [42] solved the satellite image registration problem using DDE. In [43], asynchronous DDE is implemented for the parameter estimation in biological systems.

C. Developments of ADE

The ADE has also widely studied in the DE community. However, most of the ADE variants are based on centralized DE. Researches into ADDE progress in a slow rate. As we focus on adaptive mutation strategy selection and parameters control in DDE in this article, we herein make a brief review on some ADE variants in the following two categories.

1) Parameters Control in DE: Some parameter control methods change the parameters by the fixed changing rules, while some other approaches using the feedback information from the DE search process to make parameters more adapted to the evolutionary process and more suitable for different individuals. In [32], the parameter F is adaptively adjusted according to the fitness values of individuals. Brest *et al.* [44] developed the self-adapting control parameters in DE (jDE), where the new F_i and CR_i for each individual are randomly generated in their respective ranges with probabilities τ_1 and τ_2 in each generation. Sarker *et al.* [45] proposed DE with dynamic parameters selection (DE-DPS), which consists of many combinations of parameters and the better parameters combination will be preserved and reused in the following generations. In ADE with optional external archive (JADE), proposed by Zhang and Sanderson [46], the control parameters of the individuals are updated according to their HSE. Based on JADE, Zhou *et al.* [47] introduced a modified JADE version with sorting CR, called JADE-sort, where the smaller CR value is assigned to an individual with better fitness value. Tanabe and Fukunaga [48] extended JADE by using a success-historical memory of parameter sets, called SHADE, in order to guide the generation of new control parameter values. Based on SHADE, they further improved SHADE with linear population size reduction, named L-SHADE, which later became the winner of the CEC 2014 competition on single-objective real-parameter numerical optimization [49]. After the L-SHADE, Brest *et al.* [50] presented an improved version of L-SHADE, called iL-SHADE, which improved the memory update mechanism. Based on iL-SHADE, they further developed a novel weighted version of the mutation strategy and incorporated it into the iL-SHADE, called jSO, which later became the winner of the CEC 2017 competition on single-objective real-parameter numerical optimization [51].

2) Mutation Strategy Choosing in DE: Apart from the two most frequently used mutation strategies mentioned in Section II-A, some new mutation variants are also proposed to balance the exploration and exploitation in DE. The DE/current-to-*p*best/1 mutation scheme in JADE [46], where *p*best denotes the top $p\%$ individuals, which are used to guide individuals, shown in

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,g} + F \times \left(\boldsymbol{x}^{p}_{\mathrm{best},g} - \boldsymbol{x}_{i,g}\right) + F \times \left(\boldsymbol{x}_{r1,g} - \tilde{\boldsymbol{x}}_{r2,g}\right). \quad (7)$$

In order to make use of the promising progress direction information and enhance the population diversity, the inferior solutions $\boldsymbol{x}_{i,g}$ which did not enter the next generation are recorded in an archive A, and $r2$ in (7) is randomly selected from the union, $P \cup A$, of the population and archive A.

The "DE/current-to-rand/1" strategy in [52], which involves the rotation-invariant arithmetic crossover to generate the trial vector directly. This strategy is rotation-invariant and suitable for rotated problems, shown in

$$\boldsymbol{u}_{i,g} = \boldsymbol{x}_{i,g} + \mathrm{rand} \times \left(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{i,g}\right) + F \times \left(\boldsymbol{x}_{r2,g} - \boldsymbol{x}_{r3,g}\right). \quad (8)$$

Furthermore, some adaptive mechanisms of mutation strategy have been proposed to further improve the performance of DE. For example, Qin *et al.* [31] proposed the DE with strategy adaptation (SaDE), where the mutation strategy is adaptively selected by learning from the previous experience. Wang *et al.* [29] proposed a composite DE (CoDE), which generates three trial vectors by randomly selecting three combinations from the mutation strategy pool and control parameter pool, then chooses the best one. Similarly, in DE with an ensemble of parameters and mutation strategies (EPSDE) [53], it also maintains a pool of mutation strategies and parameters of F and CR, while the successful combination will be reused in the subsequent generation.

DE with individual-dependent mechanism (IDE), proposed by Tang *et al.* [54], four mutation strategies with different searching characteristics are adaptively assigned to the superior and inferior individuals.

III. ADDE

ADDE is novel and has advantages in the following three aspects. First, ADDE uses a master–slave multipopulation distributed model, where three populations called exploration population, exploitation population, and balance population are co-evolved concurrently, which can maximize their strengths by cooperation and enhance the global optimality of DDE. Second, multiple populations can adaptively choose different mutation strategies based on the ESE information, which can fully use the feedback information from both individuals and the whole corresponding population to enable the algorithm adaptation. Third, HSE and BSI information are collected and used to adaptively update the individual parameters (F and CR) and population parameter (N), respectively. As a result, ADDE does not solely adaptively control the mutation strategy selection or parameter setting, but adaptively controls both components.

A. Master–Slave Multipopulation Distributed Model

The master–slave multipopulation distributed model is illustrated in Fig. 1. Specifically, the master node dominates three slave nodes in parallel hardware, and different populations are co-evolved concurrently on these nodes.

During the process of evolution, the master sends the entire population to these three slaves. Then, the three populations are co-evolved concurrently. Specifically, the exploitation population utilizes and updates the superior individuals with better fitness to exploit, speed up the convergence, and refine the solutions. The exploration population utilizes and updates the inferior individuals with worse fitness to explore further areas in the search space, maintain the population diversity. While the balance population utilizes and updates the normal individuals to balance diversity and convergence. Although only a part of the population is evolved in different slaves, the mutation strategies require the evolutionary information of the entire population. Therefore, the entire population is sent to each slave, as shown in Fig. 1. After the evolution, the three slaves will only send the updated superior individuals, inferior individuals, and normal individuals, respectively, to the master. The master will merge these individuals to form a new population and send the new population to these three slaves again. The process will be repeated until the termination criterion is satisfied. Noted that the master only collects the individuals from all the three slaves and sends the entire merged population to the three slaves, while the evolutionary operators, such as mutation, crossover, and selections are carried out on the slaves.

In order to define the superior, inferior, and normal individuals, we make the following two considerations. First, during the early stage of evolution, we should concentrate more on keeping population diversity, while at the later stage, we need to accelerate the convergence speed. Therefore, we propose a proportion model to gradually decrease the proportion of inferior individuals, pay more attention to exploration early while pay more attention to exploitation gradually as the algorithm progresses. Second, for the proportion of superior individuals, in order to avoid premature convergence, we keep the proportion of superior individuals small (20%) and remain unchanged. Based on these two considerations, the proportions of the superior and inferior individuals are set related to the fitness evaluations (FEs) and the maximal FEs (FEmax) as

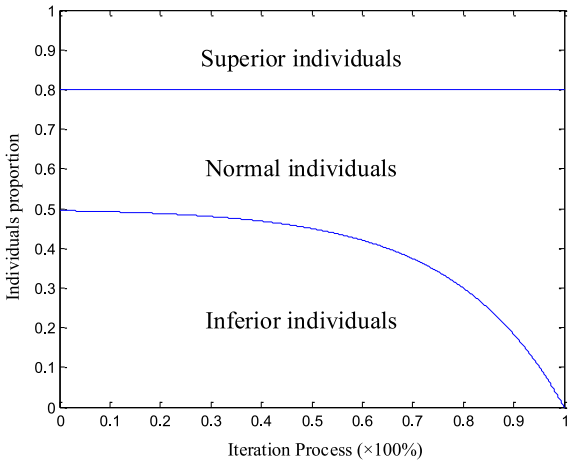


Fig. 2. Proportion model to divide the individuals.

$$\text{proportion} = \begin{cases} 0.2, & \text{Superior individuals} \\ 0.5 - 0.005 \times 10^{(2\times \text{FEs}/\text{FEmax})}, & \text{Inferior individuals.} \end{cases} \tag{9}$$

The proportions of the three kinds of subpopulation are plotted in Fig. 2, where the x-axis denotes the iteration process, and the y-axis is the proportion of the individuals ($y = 1$ means the best individual, while $y = 0$ means the worst individual).

B. Adaptive Mutation Strategy Selection Based on ESE

According to the different properties of different individuals, two schemes for adaptively choosing suitable mutation strategy for different individuals are defined and described as follows.

Scheme 1—Using DE/current-to-pbest/1 for Superior Individuals in Exploitation Population: The superior individuals with good fitness tend to exploit. As a result, the relatively greedy mutation strategy DE/current-to-*p*best/1 shown in (7) can keep their good performance on exploitation, which can accelerate the convergence speed and enhance the local searchability.

Scheme 2—Using DE/current-to-rand/1 for Inferior Individuals in Exploration Population: As for the inferior individuals, there is no need to exploit their neighborhoods because of their worse fitness. However, they can be used to explore further space. DE/current-to-rand/1 in (8) shows good diversity [52] and is suitable for these inferior individuals to explore more regions and avoid premature convergence.

As for the normal individuals, we proposed a novel ESE method to adaptively choose a suitable mutation strategy for

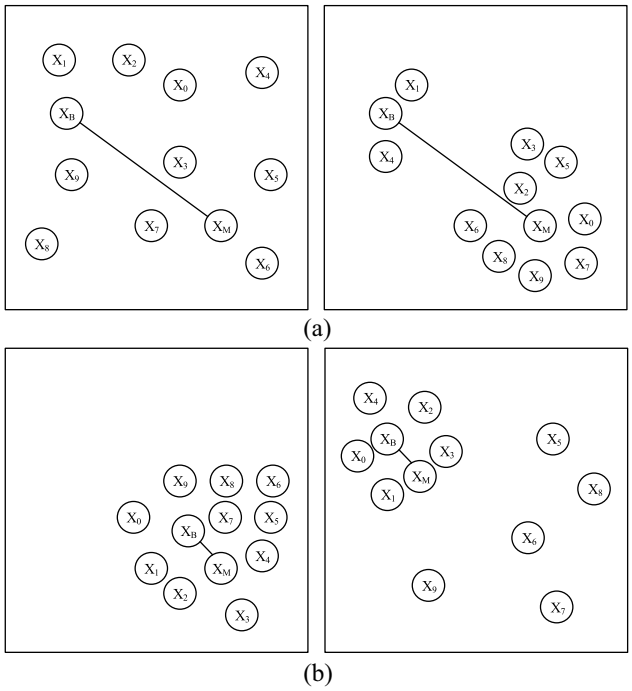


Fig. 3. Illustration of population distribution in different evolutionary states. (a) Exploration and (b) exploitation.

the normal individuals in the balance population, balance the diversity and convergence.

The ESE method has been used in the previous studies [33], [55]. However, these two methods both involve the distance computations from all the individuals, even the pairwise distance computations in [55], which are very computation costly. Here, we proposed a simple ESE method by using only two individuals.

As we all know, both the search behaviors and the population distribution vary during the evolutionary process. For instance, in the early stage, the individuals are scattered in different areas and the population distribution is dispersive, seem like the left of Fig. 3(a). However, as the evolutionary process goes on, individuals may converge to a locally or globally optimal area, seem like the left of Fig. 3(b). Such scattered or converged feedback information of the population is useful for us to understand the evolutionary process deeply. In order to use this information efficiently to control the DE process more objectively, we develop a method to estimate the evolutionary state during the DE process. The method is carried out at the beginning of every generation to estimate the current evolutionary state, by performing the following four steps.

Step 1: Denote the individual with the best fitness value and the median fitness value as $\boldsymbol{x}_B$ and $\boldsymbol{x}_M$.

Step 2: Calculate the Euclidian distance d between $\boldsymbol{x}_B$ and $\boldsymbol{x}_M$

$$d = \sqrt{\sum_{j=1}^{D} \left(x_{B,j} - x_{M,j}\right)^2} \tag{10}$$

where D is the dimension of the problem.

Step 3: Calculate the evolution factor (f), which is the percentage of d to the search space

$$f = \frac{d}{\sqrt{\sum_{j=1}^{D} \left(U_j - L_j\right)^2}}. \tag{11}$$

Step 4: Classify f into one of the two states S_1 and S_2, which represent the states of exploration and exploitation, respectively. In order to make the estimation flexible but not arbitrary, we define these two states by fuzzy sets and described as the following two cases.

Case (a)—Exploration: In this case, f is a relatively large value (e.g., larger than the threshold 0.4), somehow like Fig. 3(a). In the left of Fig. 3(a), the current population is more likely distributed in different regions, it is necessary to explore and find better regions. While in the right of Fig. 3(a), the best individual is far away from the current population, which means the current population may get trapped in the local area and should improve the diversity. We define this state as exploration state.

Case (b)—Exploitation: In contrast, a small value of f (e.g., smaller than the threshold 0.3) can be seen from Fig. 3(b). In the left of Fig. 3(b), the current population is in the same region. While in the right of Fig. 3(b), most of the individuals are around the best individual, only few individuals are distributed in other areas. At these two moments, the current population may find global optima, and it is the time to utilize the better information and accelerate the convergence speed. So, this case is likely to represent the exploitation state.

We have investigated these two thresholds in Table S.I in the supplementary material. With the above two fuzzy sets defined, we can estimate the evolutionary state of the current generation by two rules. The first rule is "unique rule." That is, if f belongs to a unique set, the current generation can be classified to the corresponding unique state. For example, if $f = 0.2$, the current state is classified to s_2 because it only belongs to s_2 which is the exploitation state. However, if f belongs to two sets, then the state is classified according to the state of the last generation and the following "stable rule." That is, the current state is still classified to state in the last generation to make the evolutionary state stable. For example, when $f = 0.33$, it can belong to both s_1 (exploration) and s_2 (exploitation). However, if the last state is s_1, we still classify the current state to s_1. While if the last state is s_2, we classify the current state to s_2.

With the evolutionary state been estimated, we can adaptively control the mutation strategy selection more pertinent for normal individuals in the balance population to match the requirements of different evolutionary states. Two more schemes and their advantages are described as follows.

Scheme 3—Using DE/current-to-rand/1 for Normal Individuals in Balance Population in Exploration State: In this state, exploring as many regions as possible and increasing the population diversity are beneficial to the population evolution. DE/current-to-rand/1 shows good diversity and is suitable in this state [52].

Scheme 4—Using DE/current-to-pbest/1 for Normal Individuals in Balance Population in Exploitation State: In

this state, we wish to accelerate the population convergence speed, so the relatively greedy mutation strategy "DE/current-to-*p*best/1" that guides the individuals toward the top $p\%$ solutions is suitable.

Based on the ESE, the three parallel populations can adaptively choose their suitable mutation strategies and run concurrently, which relieves the sensitivity of mutation strategies.

C. Adaptive Individual Parameter Settings (F and CR) Based on HSE

Good individual parameters (F and CR) can generate individuals that are more likely to survive and thus these individual parameters should be recorded and reused in the following generations. So, we use the HSE to adaptively update the individual parameters. The basic idea is to record the recent successful F and CR and use them to produce new F and CR.

As different F and CR are suitable for different strategies, successful F and CR values for each strategy are recorded and updated separately. That is, there are two different individual parameter controls for the corresponding two mutation strategies. Our individual parameter control method is an improvement of JADE [46], [56].

In JADE, the crossover rate CR_i for each individual $\boldsymbol{x}_i$ is generated according to a normal distribution with mean μ_{CR} and standard deviation 0.1, truncated to [0, 1]

$$\mathrm{CR}_i = \mathrm{rand}n_i(\mu_{\mathrm{CR}}, 0.1). \tag{12}$$

The mean μ_{CR} is updated at the end of each generation as

$$\mu_{\mathrm{CR}} = (1-c) \times \mu_{\mathrm{CR}} + c \times \mathrm{mean}_A(S_{\mathrm{CR}}) \tag{13}$$

where c is a constant set as 0.1, and mean_A is the arithmetic mean. The S_{CR} is set that stores all the successful CR values.

Similarly, the amplification factor F_i for each individual $\boldsymbol{x}_i$ is generated according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1 shown in (14), and then truncated to be 1 if $F_i > 1$ or regenerated if $F_i \leq 0$

$$F_i = \mathrm{rand}c_i(\mu_F, 0.1). \tag{14}$$

The location parameter μ_F is updated at the end of each generation as

$$\mu_F = (1-c) \times \mu_F + c \times \mathrm{mean}_{\mathrm{L}}(S_F) \tag{15}$$

where mean_L is the Lehmer mean calculated as (16). S_F is the set that stores all the successful F values

$$\mathrm{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \tag{16}$$

In order to speed up the individual parameter update process to obtain more successful F and CR, we use a new adaptive individual parameter control method, an improvement of JADE by inserting a weighting strategy. In contrast to JADE, which uses all successful F and CR values to guide individual parameter adaptation, we assign a large weight to those successful F and CR which can improve the individuals to a large degree, while a small weight to the successful F and CR which can only improve individuals a little, using the following (17) and (18). The weight for each F and CR is calculated as the normalized fitness improvement of the individual as shown in (19)

$$\mu_{\mathrm{CR}} = (1-c) \times \mu_{\mathrm{CR}} + c \times \sum_{k=1}^{|S_{\mathrm{CR}}|} (\omega_k \times S_{\mathrm{CR}}(k)) \tag{17}$$

$$\mu_F = (1-c) \times \mu_F + c \times \left(\sum_{k=1}^{|S_F|} \left(\omega_k \times S_F^2(k) \right) \right) \Big/ \left(\sum_{k=1}^{|S_F|} (\omega_k \times S_F(k)) \right) \tag{18}$$

$$\omega_k = \frac{\Delta f(k)}{\sum_{k=1}^{|S_{\mathrm{CR}}| or |S_F|} \Delta f(k)} \tag{19}$$

where $\Delta f(k) = ([|f(\boldsymbol{u}_k) - f(\boldsymbol{x}_k)|]/[|f(\boldsymbol{x}_k)|])$.

As we can see, if $\Delta f(k)$ is small, which means this pair of successful F and CR can just update the individual a little bit, then the weight ω_k will also be small correspondingly. While if $\Delta f(k)$ is large, this combination of successful F and CR can update the individual a lot, and the weight ω_k will also be large as we expected. We apply this individual parameter control method to both mutation strategies.

Note that in the "DE/current-to-rand/1" strategy, the binomial crossover operator is not applied, and CR is not required. As a result, two μ_F are needed, one is used for "DE/current-to-rand/1" strategy, and the other is used for "DE/current-to-*p*best/1." While only one μ_{CR} is needed in ADDE for "DE/current-to-*p*best/1."

D. Adaptive Population Parameter Setting (N) Based on BSI

So far, the works on adaptive population size N are relatively fewer. However, the population size is an another important control parameter in DE. The population with a small size will convergence quickly while a large size has better diversity. In order to preserve both diversity and convergence in the evolutionary process, a suitable population size is required. Thus, in our proposed ADDE, we adaptively control the population size using the feedback information of the BSI in the evolutionary process.

Denote the current population size as N. After CG generations, we calculate the absolute value of the improvement percentage of the best solution compared with that in CG generations before, shown in

$$\frac{\left| f\left(\boldsymbol{x}_{\mathrm{best},g}\right) - f\left(\boldsymbol{x}_{\mathrm{best},g-CG}\right) \right|}{\left| f\left(\boldsymbol{x}_{\mathrm{best},g-CG}\right) \right|} \tag{20}$$

where $f(\boldsymbol{x}_{\mathrm{best},g})$ is the fitness value of the current best individual and $f(\boldsymbol{x}_{\mathrm{best},g-CG})$ is the fitness value of the best individual CG generations before.

If the improvement of the best solution is larger than or equal to a threshold θ, it indicates that the population evolves well and a small population size will accelerate the convergence. As a result, NG worst individuals are removed to archive B while the remaining individuals are kept in the population. N is decreased to N–NG, and the archive B is used to store the abandoned individuals and provide new individuals for the population.

Algorithm 1 Adaptive Population Size Control

```
Begin
1.    If the improvement percentage of the best solution ≥ θ
2.      If N > N_min
3.        Move the worst NG individuals from population to archive B;
4.        N = N − NG;
5.      End If
6.    Else
7.      If N < N_max
8.        Randomly move NG individuals from archive B to population;
9.        N = N + NG;
10.     End If
11.   End If
End
```

While if the improvement is smaller than the threshold θ, or the global best solution is even not updated, it can be said that the algorithm is not performing well at the current generation. The global optimum may lie in some other regions, and the population may be trapped in local optima. As a result, new different individuals are needed to improve the diversity. *NG* individuals randomly selected from archive B are added into the current population to make the population size N equal to $N + NG$.

Note that the change percentage of the best solution will become increasingly smaller during the process of evolution. Therefore, the threshold θ should also keep reducing with the algorithm process. Here, we propose an exponential model shown in (21) to gradually reduce the threshold θ

$$\theta = 10^{-1-4\times \mathrm{FEs}/\mathrm{FEmax}}. \tag{21}$$

Two constants N_{min} and $N_{\max}$ are used to control the bound of the population size. Besides, since the individuals stored and taken are all from archive B, so the sum of the population size N and the size of archive B remains constant in the entire evolution process. The detailed pseudocode of the population size control is illustrated in Algorithm 1.

Combining with all the adaptive control components, the pseudocode of ADDE is presented in Algorithm 2.

Algorithm 2: ADDE algorithm

```
Process of MASTER in ADDE
Begin
1. Initialize population randomly; g=0; FEs=0;
2. Evaluate the population;
3. FEs=FEs+N;
4. While FEs≤FEmax do
5.    Evolutionary state estimation (ESE);
6.    Send the whole population to three SLAVEs;
7.    Receive the updated individuals from three SLAVEs;
8.    Merges the updated individuals to form a new population;
9.    Adaptive individual parameter settings (F and CR) based on HSE using (18)-(20);
10.   If g%CG==0
11.      Adaptive population parameter setting (N) based on BSI using Algorithm 1;
12.   End If
13.   FEs=FEs+N;
14.   g=g+1;
15. End While
End

Process of SLAVE 1 in ADDE
//Exploration population
Begin
1. Receive the population from the MASTER;
2. Choose the DE/current-to-rand/1 mutation strategy;
3. Generate F_i using (15);
4. Updated the inferior individuals in the exploration population;
5. Send the updated inferior individuals to MASTER.
End

Process of SLAVE 2 in ADDE
//Balance population
Begin
1.  Receive the population from the MASTER;
2.  If Exploration State
3.     Choose the DE/current-to-rand/1 mutation strategy;
4.     Generate F_i using (15);
5.  End If
6.  If Exploitation State
7.     Choose the DE/current-to-pbest/1 mutation strategy;
8.     Generate CR_i and F_i using (13) and (15);
9.  End If
10. Updated the normal individuals in the balance population.
11. Send the updated normal individuals MASTER.
End

Process of SLAVE 3 in ADDE
//Exploitation population
Begin
1. Receive the population from the MASTER;
2. Choose the DE/current-to-pbest/1 mutation strategy;
3. Generate CR_i and F_i using (13) and (15);
4. Updated the superior individuals in the exploitation population.
5. Send the updated superior individuals MASTER.
End
```

E. Complexity Analysis

Herein, we denote the population size, the dimension of problem, and the change quantity of population size as N, D, and NG, respectively. First, in the initialization, the time complexity of ADDE is $O(N \times D) + O(N)$, which is obtained by steps 1–3 in MASTER process of Algorithm 2. Then, as for individual evolution in ADDE, ESE is first executed, whose time complexity is $O(D) + O(N)$, obtained by (10) and (11). Then, ADDE will send the entire population from MASTER to three SLAVEs, where three populations called exploration population, exploitation population, and balance population are co-evolved concurrently by updating the superior, inferior, and normal individuals, respectively. The time complexity in defining the superior, inferior, and normal individuals is $O(N \times \log(N))$, obtained by (9) and Fig. 2, while the sum of time complexity in the process of evolution on three SLAVEs is $O(N \times D)$, since the sum of updated individuals is N. After that, the adaptive individual parameter settings (F and CR) based on HSE is executed on MASTER, with the time complexity of $O(N)$, which is obtained by (17)–(19). Next, the adaptive population parameter setting (N) based on BSI is executed on MASTER, with the time complexity of $O(NG \times D)$, which is obtained by Algorithm 1. As a result, the overall time complexity of ADDE is $O(N \times D) + O(N \times \log(N))$.

IV. Experimental Results

A. Benchmark Functions and Experimental Setup

In this section, all the 30 functions proposed in the CEC 2014 test suite [57] are used to evaluate the performance of ADDE. These functions can be classified into four groups. The first group includes the first three functions f_1–f_3 which are unimodal functions. The second group includes the next 13 functions f_4–f_{16}which are multimodal functions with a huge number of local optima. The third group includes the next six functions f_{17}–f_{22} that are hybrid functions. The last group includes the last eight functions f_{23}–f_{30} that are composition functions, which are much more complex and make our test suite more comprehensive and convincing.

The maximum and minimum population size $N_{\max}$ and N_{min} of ADDE are set to $10 \times D$ and $2 \times D$. The initial population of ADDE is set as $N_{\max}$. The change quantity of population size NG is set as $0.4\times D$, while the time span CG is set as 30 generations. The initial values of μ_F and μ_{CR} of ADDE are both set as 0.5. The size of archive A has a certain threshold *Aup*, which is 2.5 times

TABLE I
COMPARISON RESULTS BETWEEN ADDE AND STATE-OF-THE-ART DDES ON 30-D PROBLEMS

fun	ADDE	IBDDE	Cloudde	AsAMP-dDE	DDE-SD	DDEM-RCA
	Mean±Std	Mean±Std	Mean±Std	Mean±Std	Mean±Std	Mean±Std
f_1	**0.00E+00±0.00E+00**	9.97E+06±1.05E+07(+)	2.58E+05±1.78E+06(+)	1.98E+05±4.51E+05(+)	6.70E+05±4.68E+06(+)	1.61E+06±4.20E+05(+)
f_2	**0.00E+00±0.00E+00**	2.60E+08±9.23E+08(+)	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	5.18E+03±5.72E+03(+)	6.99E+02±1.91E+02(+)
f_3	**0.00E+00±0.00E+00**	5.15E+03±5.35E+03(+)	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	3.93E-01±9.67E-09(+)	1.47E-02±4.09E-03(+)
f_4	**0.00E+00±0.00E+00**	1.40E+02±4.11E+01(+)	3.50E+01±1.68E+02(+)	3.55E+01±4.07E+01(+)	7.79E+01±3.32E+01(+)	6.99E+01±1.24E+01(+)
f_5	**2.03E+01±2.85E-02**	2.06E+01±1.05E-01(+)	2.05E+01±3.07E-01(+)	2.08E+01±1.83E-01(+)	2.05E+01±8.00E-02(+)	2.09E+01±5.32E-02(+)
f_6	**1.85E-03±3.10E-02**	2.38E+01±2.60E+00(+)	5.93E+00±1.46E+01(+)	2.42E+01±4.33E+00(+)	1.38E+01±2.64E+00(+)	9.31E+00±2.50E+01(+)
f_7	**0.00E+00±0.00E+00**	1.87E+00±4.97E+00(+)	**0.00E+00±0.00E+00(≈)**	1.79E-02±1.89E-02(+)	2.37E-02±1.48E-02(+)	8.03E-04±2.97E-04(+)
f_8	**0.00E+00±0.00E+00**	7.38E+01±2.63E+01(+)	**0.00E+00±0.00E+00(≈)**	7.01E+01±2.19E+01(+)	1.71E+01±9.73E+00(+)	1.70E+02±1.22E+01(+)
f_9	**1.40E+01±2.87E+00**	1.07E+02±3.21E+01(+)	3.93E+01±5.97E+01(+)	1.02E+02±2.60E+01(+)	4.90E+01±1.50E+01(+)	1.92E+02±9.12E+00(+)
f_{10}	**3.39E-01±2.22E-01**	1.53E+03±6.54E+02(+)	9.52E+00±2.40E+01(+)	1.85E+03±6.17E+02(+)	3.23E+02±2.78E+02(+)	5.14E+03±3.19E+02(+)
f_{11}	**1.72E+03±3.27E+02**	3.63E+03±7.64E+02(+)	2.93E+03±3.05E+03(+)	4.87E+03±1.11E+03(+)	2.79E+03±5.48E+02(+)	6.97E+03±2.77E+02(+)
f_{12}	4.06E-01±6.44E-02	5.55E-01±3.35E-01(+)	3.90E-01±8.94E-01(-)	1.24E+00±6.19E-01(+)	**2.86E-01±1.04E-01(-)**	2.43E+00±2.70E-01(+)
f_{13}	**1.42E-01±1.53E-02**	4.17E-01±1.12E-01(+)	3.15E-01±2.74E-01(+)	4.01E-01±1.08E-02(+)	2.43E-01±5.45E-02(+)	4.40E-01±5.71E-02(+)
f_{14}	**2.23E-01±4.22E-02**	3.07E-01±6.31E-02(+)	2.84E-01±1.95E-01(+)	3.33E-01±1.60E-01(+)	2.43E-01±4.92E-02(+)	2.74E-01±3.67E-02(+)
f_{15}	**2.52E+00±6.21E-01**	2.47E+02±2.26E+02(+)	5.32E+00±1.40E+01(+)	1.62E+01±7.74E+00(+)	5.98E+00±1.71E+00(+)	1.68E+01±1.17E+00(+)
f_{16}	**9.49E+00±2.77E-01**	1.13E+01±6.58E-01(+)	1.06E+01±3.07E+00(+)	1.24E+01±7.11E-01(+)	1.09E+01±5.17E-01(+)	1.29E+01±1.45E-01(+)
f_{17}	**4.57E+02±2.13E+02**	1.04E+06±7.73E+05(+)	4.32E+03±2.17E+04(+)	6.35E+04±1.58E+05(+)	3.49E+04±3.42E+04(+)	1.94E+03±1.78E+02(+)
f_{18}	**1.85E+01±6.36E+00**	5.69E+03±5.92E+03(+)	6.94E+01±2.54E+02(+)	1.07E+03±3.10E+03(+)	1.27E+02±2.65E+02(+)	7.49E+01±8.22E+00(+)
f_{19}	**3.64E+00±5.23E-01**	5.46E+01±4.60E+01(+)	4.34E+00±6.80E+00(+)	1.05E+01±1.13E+01(+)	6.34E+00±1.45E+00(+)	5.67E+00±3.85E-01(+)
f_{20}	**5.22E+00±1.19E+00**	1.14E+04±9.85E+03(+)	6.53E+01±3.47E+02(+)	1.62E+02±8.47E+01(+)	1.77E+02±1.58E+02(+)	4.87E+01±3.65E+00(+)
f_{21}	**7.69E+01±7.23E+01**	3.53E+05±3.31E+05(+)	1.05E+03±5.04E+03(+)	3.77E+03±9.87E+03(+)	5.75E+03±5.90E+03(+)	1.04E+03±1.51E+02(+)
f_{22}	**2.86E+01±4.60E+00**	6.02E+02±1.93E+02(+)	1.83E+02±5.94E+02(+)	5.74E+02±1.72E+02(+)	2.70E+02±1.19E+02(+)	2.42E+02±1.45E+02(+)
f_{23}	**3.15E+02±0.00E+00**	3.17E+02±2.22E+00(+)	**3.15E+02±8.14E-11(≈)**	**3.15E+02±9.47E-13(≈)**	**3.15E+02±4.42E-08(≈)**	**3.15E+02±6.23E-06(≈)**
f_{24}	2.23E+02±4.76E+00	2.51E+02±8.01E+00(+)	2.25E+02±5.61E-01(+)	2.41E+02±9.18E+00(+)	2.25E+02±1.13E-01(+)	**2.05E+02±5.18E+00(-)**
f_{25}	**2.03E+02±2.21E-01**	2.16E+02±4.87E+00(+)	2.04E+02±3.69E+00(+)	2.43E+02±6.41E+00(+)	2.05E+02±1.87E+00(+)	2.04E+02±3.70E+00(+)
f_{26}	**1.00E+02±2.68E-02**	1.27E+02±4.49E+01(+)	**1.00E+02±2.98E-01(+)**	1.01E+02±1.72E-01(+)	**1.00E+02±9.12E-01(+)**	**1.00E+02±4.21E-01(+)**
f_{27}	**3.00E+02±0.00E+00**	8.28E+02±2.75E+02(+)	3.95E+02±1.56E+02(+)	7.05E+02±2.85E+02(+)	4.21E+02±6.22E+01(+)	3.91E+02±2.68E+01(+)
f_{28}	8.19E+02±1.69E+01	1.48E+03±3.46E+02(+)	8.36E+02±2.26E+02(+)	1.17E+03±3.00E+02(+)	9.40E+02±5.84E+01(+)	**8.15E+02±2.08E+01(≈)**
f_{29}	**7.17E+02±2.68E+00**	2.81E+06±4.17E+06(+)	1.25E+03±5.41E+03(+)	2.93E+06±4.39E+06(+)	1.30E+03±2.92E+02(+)	8.52E+02±4.75E+01(+)
f_{30}	**8.61E+02±4.96E+02**	1.73E+04±1.89E+04(+)	1.92E+03±5.01E+03(+)	3.17E+03±2.16E+03(+)	1.97E+03±7.24E+02(+)	1.33E+03±2.11E+02(+)
+(ADDE is significantly better)		30	24	27	28	27
-(ADDE is significantly worse)		0	1	0	1	1
≈		0	5	3	1	2

of N. The parameter p in (7), which determines the greediness of the mutation strategy, is set as 10%. We compare ADDE with five DDE variants, including IBDDE [22], Cloudde [23], AsAMP-dDE [25], DDE-SD [38], and DDEM-RCA [39]. Moreover, we also compare ADDE with nine ADE variants, including CoDE [29], SaDE [31], jDE [44], DE-DPS [45], JADE [46], JADE-sort [47], SHADE [48], EPSDE [53], and IDE [54]. To have a reliable and fair comparison, the parameter configurations (including the values of N, F, CR, and other additional parameters) in all the competitor algorithms are set the same as suggested in their original papers. The FEmax is set as $10\,000\times D$ in all algorithms.

Due to the stochastic character of the algorithms, all the algorithms run 30 times independently and the mean error value is calculated to evaluate the performance of the algorithms. Here, the error value smaller than 10^{-8} will be taken as 0, which is the numerical zero on computer arithmetic. For clarity, the results of the best algorithms are marked in boldface. In addition, Wilcoxon's rank-sum test [58] at $\alpha = 0.05$ is tested to evaluate the statistical significance of the results between different algorithms. The symbols $+$, $\approx$, and $-$ indicate ADDE performs significantly better ($+$), similarly ($\approx$), or worse ($-$) when compared with the corresponding algorithm.

B. Comparison With DDEs on 30D Problems

We first compare ADDE with the 5 state-of-the-art DDE variants on 30D benchmark functions to evaluate its overall performance. The detailed experimental results are listed in Table I. From Table I, we can see the following.

For the unimodal functions f_1–f_3, only ADDE can always find the global optimal solution on all the three functions. Moreover, ADDE performs significantly better than other DDE variants, especially on f_1.

For the multimodal functions f_4–f_{16}, ADDE achieves the best performance on almost all the functions, except on f_{12}. Only DDE-SD performs slightly better than ADDE on f_{12}, while other algorithms are all dominated by ADDE on all the functions.

For the hybrid functions f_{17}–f_{22}, ADDE still keeps its superiority and outperforms all the DDE variants on all these functions.

For the composition functions f_{23}–f_{30}, DDEM-RCA achieves the best performance on f_{24}, while almost all the competitors achieve the similar performance on f_{23}. However, ADDE still performs significantly better than other DDE variants on all the other functions.

Overall, ADDE performs better than IBDDE, Cloudde, AsAMP-dDE, DDE-SD, and DDEM-RCA on 30, 24, 27, 28, and 27 functions, respectively. Conversely, Cloudde, DDE-SD, and DDEM-RCA can only surpass ADDE on 1 function. IBDDE and AsAMP-dDE cannot outperform ADDE on any function in our test. As a result, ADDE achieves the best performance on these functions.

To further study the evolutionary behavior of different algorithms, we draw their convergence curves to observe their evolutionary processes. Besides, in order to make our

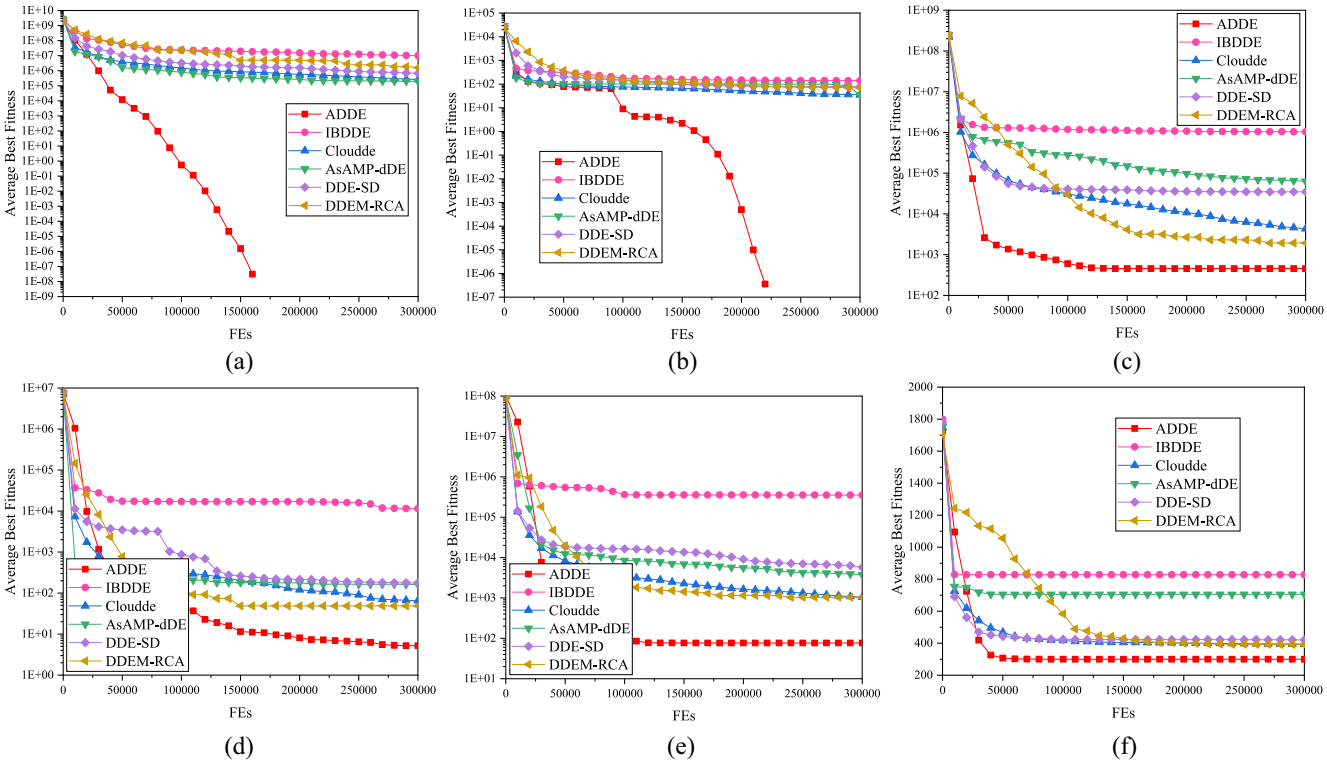


Fig. 4. Convergence curves of ADDE and other state-of-the-art DDE variants on six representative functions. (a) Unimodal function f_1, (b) multimodal function f_4, (c) hybrid function f_{17}, (d) hybrid function f_{20}, (e) hybrid function f_{21}, and (f) composition function f_{27}.

comparison more convincing, we choose several benchmark functions from all the three groups. Here, we select unimodal function f_1, multimodal function f_4, hybrid functions f_{17}, f_{20}, and f_{21}, and composition function f_{27} as the representative instances. The convergence curves of these algorithms on the 6 selected benchmark functions are plotted in Fig. 4. From Fig. 4(a), we can see that ADDE finds the global optimal solution quickly while other algorithms occur stagnation in the early stage or evolve slower on unimodal function f_1. Similar phenomenon can also be observed on the multimodal function f_4 in Fig. 4(b). On hybrid functions f_{17}, f_{20}, and f_{21}, shown in Fig. 4(c)–(e), IBDDE converges slower while other compared DDE variants can achieve a similar performance. Moreover, ADDE obtains the highest accuracy when compared with all the other algorithms. For composition function f_{27} in Fig. 4(f), ADDE converges a little slower in the early stage of the evolutionary process, but obtains a better solution compared with other algorithms in the later stage and finally obtains the best result. Overall, ADDE generally outperforms other DDE variants.

C. Comparison With ADEs on 30D Problems

To further verify the effectiveness of ADDE, here, ADDE is compared with nine state-of-the-art ADE variants on 30D benchmark functions to evaluate its overall performance. Table II summarizes the comparison results between ADDE and other ADE algorithms, while the detailed comparison results are shown in Table S.II in the supplementary material for saving space. From Table II and Table S.II in the supplementary material, we can see the following.

For the unimodal functions f_1–f_3, most of the ADE variants can achieve the similar performance to ADDE on f_2 and f_3. However, ADDE performs significantly better than other ADE variants and obtains the global optimal solution on f_1.

For the multimodal functions f_4–f_{16}, we can see that ADDE, JADE-sort, and SHADE generally outperform other ADE variants. Even both JADE-sort and SHADE dominates ADDE on f_5, f_{11}, f_{12}, and f_{16}, ADDE still achieves higher accuracy than JADE-sort and SHADE on f_4, f_6, f_8, f_{10}, f_{14}, and f_6, f_7, f_{13}, f_{14}, respectively. Such results indicate that ADDE can keep population diversity and avoid premature convergence.

For the hybrid functions f_{17}–f_{22}, ADDE gradually shows its tremendous superiority. ADDE obtain the best performance on almost all these test functions, except on f_{18} and f_{19}.

For the composition functions f_{23}–f_{30}, there is no ADE variant which can always obtain the best performance. For instance, EPSDE performs best on f_{25}, f_{28}, and f_{29}, while IDE achieve the best performance on f_{30}. However, ADDE still keeps its dominant position on most functions, such as f_{24}, f_{26}, and f_{27}. Such observation fully shows the superiority of ADDE when solving more complex problems.

Overall, ADDE performs better than CoDE, SaDE, jDE, DE-DPS, JADE, JADE-sort, SHADE, EPSDE, and IDE on 19, 27, 20, 25, 18, 16, 16, 20, and 13 functions, respectively. Conversely, CoDE, SaDE, jDE, JADE, JADE-sort, SHADE, EPSDE, and IDE can only surpass ADDE on 5, 1, 2, 4, 6, 5, 5, and 8 functions, respectively. DE-DPS cannot outperform

TABLE II
SUMMARIZED RESULTS BETWEEN ADDE AND STATE-OF-THE-ART ADES ON 30-D PROBLEMS

ADDE	CoDE	SaDE	jDE	DE-DPS	JADE	JADE-sort	SHADE	EPSDE	IDE
+(ADDE is significantly better)	19	27	20	25	18	16	16	20	13
-(ADDE is significantly worse)	5	1	2	0	4	6	5	5	8
≈	6	2	8	5	8	8	9	5	9

TABLE III
SUMMARIZED RESULTS BETWEEN ADDE AND STATE-OF-THE-ART DDES ON 100-D PROBLEMS

ADDE	IBDDE	Cloudde	AsAMP-dDE	DDE-SD	DDEM-RCA
+(ADDE is significantly better)	30	21	27	24	24
-(ADDE is significantly worse)	0	5	2	3	4
≈	0	4	1	3	2

TABLE IV
SUMMARIZED RESULTS BETWEEN ADDE AND STATE-OF-THE-ART ADES ON 100-D PROBLEMS

ADDE	CoDE	SaDE	jDE	DE-DPS	JADE	JADE-sort	SHADE	EPSDE	IDE
+(ADDE is significantly better)	21	25	17	23	19	17	18	21	17
-(ADDE is significantly worse)	6	3	8	4	5	7	5	7	10
≈	3	2	5	3	6	6	7	2	3

ADDE on any function in our test. As a result, ADDE achieves the best performance on these functions.

We further draw the convergence curves of these algorithms to observe their evolutionary processes. Similarly, we also select unimodal function f_1, multimodal function f_4, hybrid functions f_{17}, f_{20}, f_{21}, and composition function f_{27} as the representative instances. The convergence curves of these algorithms on the 6 selected benchmark functions are plotted in Fig. S1 in the supplementary material due to the page limitation. From Fig. S1(a) in the supplementary material, we can see that ADDE converges to better solution quickly while other algorithms occur stagnation in the early stage or evolve slower on unimodal function f_1. In Fig. S1(b) in the supplementary material, only ADDE, JADE, and SHADE can converge to the optimal solution on multimodal function f_4. While in Fig. S1(c) in the supplementary material, most of the competitors can achieve similar performance on hybrid function f_{17}, however, ADDE is still more accurate than other algorithms. Similar phenomenon can also be observed on the functions f_{20} and f_{21} in Fig. S1(d) and (e) in the supplementary material. For composition function f_{27} in Fig. S1(f) in the supplementary material, ADDE appears a short stagnation in the middle of an evolutionary process, but obtains the best solution among all the compared algorithms in the following stage. Overall, ADDE outperforms other ADE variants.

D. Scalability of ADDE on 100D Problems

In order to investigate the scalability of ADDE, we further compare the performance of ADDE with these DDE and ADE variants on f_1–f_{30} with 100D. The summarized results on 100D are shown in Tables III and IV, while the detail experimental results on 100D can be seen in Tables S.III and S.IV in the supplementary material due to the space limitation. As we can see, with the increasing dimensions, the performance of many algorithms is greatly deteriorated. Besides, Cloudde, SHADE, and IDE may be more suitable to solve multimodal problems, while CoDE, jDE, and JADE-sort perform relatively better on hybrid functions. Even so, ADDE still keeps fast convergence on unimodal problems while maintains the population diversity on multimodal or other complex problems. It performs better than IBDDE, Cloudde, AsAMP-dDE, DDE-SD, and DDEM-RCA on 30, 21, 27, 24, and 24 functions, respectively and performs better than CoDE, SaDE, jDE, DE-DPS, JADE, JADE-sort, SHADE, EPSDE, and IDE on 21, 25, 17, 23, 19, 17, 18, 21, and 17 functions, respectively. While Cloudde, DDE-SD, and DDEM-RCA can only surpass ADDE on 5, 2, 3, and 4 functions, respectively, CoDE, SaDE, jDE, DE-DPS, JADE, JADE-sort, SHADE, EPSDE, and IDE can only surpass ADDE on 6, 3, 8, 4, 5, 7, 5, 7, and 10 functions, respectively. IBDDE cannot outperform ADDE on any function in our test. These results demonstrate that ADDE can also remain good performance when the dimension increases to 100.

E. Comparison With the Winner of the CEC 2014 Competition

To further demonstrate the superiority of ADDE, in this section, we compare ADDE with the winner of the CEC 2014 competition, L-SHADE [49], and its variant jSO [51]. The codes are downloaded from http://www.ntu.edu.sg/home/epnsugan/. The detailed comparison results between ADDE, L-SHADE, and jSO are listed in Table V. The best results are highlighted in boldface. From Table V, we can see the following.

1) On 30D problems, ADDE can achieve similar performance with L-SHADE and jSO when solving the unimodal problems.
2) L-SHADE and jSO have the superiority on the multimodal and hybrid problems. However, ADDE has its advantage on the composition problems and dominates L-SHADE on f_{24}, f_{28}, and f_{30}, dominates jSO on f_{24} and f_{28}.

TABLE V
COMPARISON RESULTS BETWEEN ADDE AND WINNER OF THE CEC 2014 COMPETITION

fun	30D			fun	100D		
	ADDE	L-SHADE	jSO		ADDE	L-SHADE	jSO
	Mean±Std	Mean±Std	Mean±Std		Mean±Std	Mean±Std	Mean±Std
f_1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_1	1.94E+05±6.56E+04	1.52E+05±5.36E+04(-)	**1.31E+05±4.75E+04(-)**
f_2	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_2	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**
f_3	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_3	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**
+(ADDE is significantly better)		0	0	+(ADDE is significantly better)		0	0
-(ADDE is significantly worse)		0	0	-(ADDE is significantly worse)		1	1
≈		3	3	≈		2	2
f_4	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_4	**1.55E+02±4.40E+01**	1.62E+02±2.66E+01(+)	1.63E+02±2.71E+01(+)
f_5	2.03E+01±2.85E-02	**2.01E+01±2.29E-02(-)**	2.08E+01±2.68E-02(+)	f_5	2.09E+01±1.34E-02	**2.05E+01±3.34E-02(-)**	2.09E+01±3.30E-01(≈)
f_6	**1.85E-03±3.10E-02**	1.83E-02±9.34E-02(+)	9.40E-03±5.06E-02(+)	f_6	2.35E+01±1.66E+00	9.82E+00±2.39E+00(-)	**3.51E+00±1.81E+00(-)**
f_7	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_7	**0.00E+00±0.00E+00**	2.47E-04±1.33E-03(+)	**0.00E+00±0.00E+00(≈)**
f_8	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00(≈)**	**0.00E+00±0.00E+00(≈)**	f_8	**0.00E+00±0.00E+00**	1.19E-02±6.78E-03(+)	4.13E-03±1.47E-03(+)
f_9	1.40E+01±2.87E+00	**6.67E+00±1.51E+00(-)**	9.00E+00±1.77E+00(-)	f_9	6.74E+01±6.81E+01	**3.22E+01±4.78E+00(-)**	4.10E+01±7.68E+00(-)
f_{10}	3.39E-01±2.22E-01	**1.67E-02±1.89E-02(-)**	1.11E+00±8.43E-01(+)	f_{10}	4.62E+02±1.17E+02	**2.55E+01±5.24E+00(-)**	9.02E+01±2.10E+01(-)
f_{11}	1.72E+03±3.27E+02	1.26E+03±1.63E+02(-)	**1.24E+03±2.58E+02(-)**	f_{11}	1.63E+04±1.03E+03	1.09E+04±5.11E+02(-)	**9.91E+03±6.56E+02(-)**
f_{12}	4.06E-01±6.44E-02	**1.58E-01±2.53E-02(-)**	3.28E-01±2.97E-02(-)	f_{12}	1.27E+00±1.01E-01	**4.24E-01±3.90E-02(-)**	4.53E-01±4.90E-02(-)
f_{13}	1.42E-01±1.53E-02	**1.19E-01±1.35E-02(-)**	1.41E-01±2.16E-02(≈)	f_{13}	3.04E-01±3.93E-02	**2.43E-01±1.37E-02(-)**	3.16E-01±5.33E-02(+)
f_{14}	**2.23E-01±4.22E-02**	2.25E-01±2.67E-02(≈)	2.44E-01±4.04E-02(+)	f_{14}	3.66E-01±2.67E-02	**2.22E-01±1.04E-02(-)**	2.21E+02±4.45E+01(+)
f_{15}	2.52E+00±6.21E-01	**2.13E+00±2.77E-01(-)**	2.26E+00±3.01E-01(-)	f_{15}	**1.08E+01±1.84E+00**	1.64E+01±1.03E-01(+)	1.57E+01±1.59E+00(+)
f_{16}	9.49E+00±2.77E-01	**8.45E+00±3.80E-01(-)**	8.95E+00±9.08E-01(-)	f_{16}	4.23E+01±4.19E-01	3.94E+01±4.42E-01(-)	**3.89E+01±6.26E-01(-)**
f_{17}	4.57E+02±2.13E+02	2.12E+02±1.03E+02(-)	**6.84E+01±2.02E+01(-)**	f_{17}	4.79E+03±7.99E+02	4.28E+03±7.63E+02(-)	**3.65E+03±6.77E+02(-)**
f_{18}	1.85E+01±6.36E+00	5.58E+00±2.43E+00(-)	**2.37E+00±1.43E+00(-)**	f_{18}	**2.12E+02±3.34E+01**	2.21E+02±1.24E+01(+)	2.17E+02±2.00E+01(+)
f_{19}	3.64E+00±5.23E-01	3.74E+00±5.71E-01(-)	**2.06E+00±6.51E-01(-)**	f_{19}	9.92E+01±3.43E+00	9.54E+01±2.38E+00(-)	**9.10E+01±1.17E+00(-)**
f_{20}	5.22E+00±1.19E+00	2.97E+00±1.42E+00(-)	**2.15E+00±1.01E+00(-)**	f_{20}	3.46E+02±6.19E+01	1.59E+02±5.12E+01(-)	**4.90E+01±1.06E+01(-)**
f_{21}	7.69E+01±7.23E+01	8.06E+01±6.51E+01(+)	**1.24E+01±8.97E+00(-)**	f_{21}	2.43E+03±5.84E+02	2.14E+03±5.73E+02(-)	**9.07E+02±2.08E+02(-)**
f_{22}	2.86E+01±4.60E+00	**2.50E+01±2.50E+01(≈)**	3.75E+01±3.52E+01(+)	f_{22}	1.09E+03±2.40E+02	1.14E+03±1.30E+01(≈)	**1.00E+03±2.32E+02(≈)**
+(ADDE is significantly better)		2	5	+(ADDE is significantly better)		5	6
-(ADDE is significantly worse)		12	10	-(ADDE is significantly worse)		13	10
≈		5	4	≈		1	3
f_{23}	**3.15E+02±0.00E+00**	**3.15E+02±5.68E-14(≈)**	**3.15E+02±5.68E-14(≈)**	f_{23}	**3.48E+02±0.00E+00**	**3.48E+02±1.80E-13(≈)**	**3.48E+02±2.37E-13(≈)**
f_{24}	2.23E+02±4.76E+00	2.24E+02±1.14E+00(+)	**2.26E+02±9.59E+00(+)**	f_{24}	**3.88E+02±2.51E+00**	3.94E+02±2.71E+00(+)	3.98E+02±6.26E+00(+)
f_{25}	**2.03E+02±2.21E-01**	**2.03E+02±3.78E-02(≈)**	**2.03E+02±1.76E-02(≈)**	f_{25}	**2.00E+02±3.07E-13**	**2.00E+02±2.01E-13(≈)**	**2.00E+02±2.66E-13(≈)**
f_{26}	**1.00E+02±2.68E-02**	**1.00E+02±1.77E-02(≈)**	**1.00E+02±2.73E-02(≈)**	f_{26}	**1.90E+02±3.15E+01**	2.00E+02±4.77E-13(+)	2.00E+02±3.65E-13(+)
f_{27}	**3.00E+02±0.00E+00**	**3.00E+02±2.40E-13(≈)**	**3.00E+02±1.17E-13(≈)**	f_{27}	5.67E+02±4.25E+01	**3.79E+02±3.14E+01(-)**	4.25E+02±3.98E+01(-)
f_{28}	**8.19E+02±1.69E+01**	8.40E+02±1.11E+01(+)	8.29E+02±1.56E+01(+)	f_{28}	2.22E+03±5.10E+01	**2.20E+02±5.39E+01(≈)**	2.21E+02±4.44E+01(≈)
f_{29}	7.17E+02±2.68E+00	7.16E+02±2.83E+00(≈)	**7.15E+02±2.07E+00(≈)**	f_{29}	1.09E+03±1.82E+02	**8.32E+02±1.19E+02(-)**	1.00E+03±3.77E+02(≈)
f_{30}	8.61E+02±4.96E+02	1.38E+03±6.37E+02(+)	**6.10E+02±2.15E+02(-)**	f_{30}	**6.97E+03±9.10E+02**	8.26E+03±1.07E+03(+)	7.06E+03±8.23E+02(≈)
+(ADDE is significantly better)		3	2	+(ADDE is significantly better)		3	2
-(ADDE is significantly worse)		0	1	-(ADDE is significantly worse)		2	1
≈		5	5	≈		3	5

TABLE VI
SUMMARIZED RESULTS BETWEEN ADDE AND OTHER ALGORITHMS ON REAL-WORLD APPLICATION PROBLEMS

ADDE	IBDDE	Cloudde	AsAMP-dDE	DDE-SD	DDEM-RCA	CoDE	SaDE
+	19	17	19	14	19	13	15
-	0	2	0	2	0	3	2
≈	3	3	3	6	3	6	5
jDE	DE-DPS	JADE	JADE-sort	SHADE	EPSDE	IDE	jSO
15	19	15	9	12	16	8	5
0	0	1	1	2	1	2	5
7	3	6	12	8	5	12	12

3) When the dimension increases, ADDE still keeps its superiority on composition problems and dominates L-SHADE on f_{24}, f_{26}, and f_{30}, dominates jSO on f_{24} and f_{26}.

Therefore, ADDE, L-SHADE, and jSO have their different advantages on different problems.

F. Comparison With on Real-World Application Problems

We further test the performance of ADDE on a suite of real-world application problems from the CEC 2011 test suite [59] to verify its practicability. The summarized comparison results between ADDE and other DDE and ADE algorithms are shown in Table VI, while the detailed comparison results are shown in Table S.V in the supplementary material for saving space. Moreover, the jSO algorithm is adopted for further comparison. The FEmax is set as 150 000 in all algorithms, as required in [59]. From Table VI and Table S.V in the supplementary material, we can see that ADDE is significantly superior to IBDDE, Cloudde, AsAMP-dDE, DDE-SD, DDEM-RCA, CoDE, SaDE, jDE, DE-DPS, JADE, SHADE, and EPSDE on at least 10 problems, while none of these algorithms can significantly surpass ADDE on more than 3 problems. Moreover, ADDE significantly outperforms JADE-sort and IDE on 9 and 8 problems, respectively, while it is significantly beaten by them on only 1 and 2 problems, respectively. When compared ADDE with jSO, both the two algorithms have very similar performance. The results in Table VI show that ADDE and jSO play even and significantly outperform

each other on 5 problems, while the results in Table S.V in the supplementary material further show that they are both the two most superior algorithms among all the competitors in solving these problems. As a result, ADDE has a very good practicability and can be applied to solve the real-world application problems.

G. Effects of ADDE Components

The main components of ADDE are the strategy adaptation selection and parameter adaptation control (F, CR, and N). In this section, we will discuss the property and effect of each component.

1) Strategy Adaptation Selection: First, in order to investigate the effect of strategy adaptation selection, we compare ADDE with two ADDE variants with only one mutation strategy (i.e., DE/current-to-rand/1 and DE/current-to-$p$best/1) and one ADDE variant with random mutation strategy (50% chance to use both mutation strategy). We denote the ADDE variant with only one strategy of DE/current-to-rand/1 and DE/current-to-$p$best/1 as ADDE-rand and ADDE-$p$best, respectively and denote the ADDE variant with random mutation strategy as ADDE-50%. The detailed results of the comparisons are listed in Table S.VI in the supplementary material. The effectiveness of DE/current-to-rand/1 has been verified when it was applied to solve rotated problems [52], [60], because it can maintain good diversity of the population. It also performs well on the hybrid function f_{17}. However, it also slows the convergence speed on some other functions, as can be seen from the Table S.VI in the supplementary material. Moreover, DE/current-to-$p$best/1 can accelerate the convergence speed by utilizing the top p% individuals to guide the flying of individuals, which is very useful to solve unimodal problems f_2 and f_3, but it also gets trapped in local optima on some other problems.

As we can see, not a single mutation strategy is attractive, and only an appropriate mutation strategy is beneficial. However, without any prior knowledge about the test function, ADDE still performs better than ADDE-$p$best and ADDE-rand on 18 and 26 functions. ADDE-50% can outperform ADDE-$p$best and ADDE-rand on many functions, which indicates the combination of two mutation strategy is helpful for algorithm. However, ADDE-50% is still much worse than ADDE, which can only dominate ADDE on 1 function but dominated by ADDE on 20 functions. Therefore, the adaptive mutation strategy in ADDE is intelligent and useful to select the suitable mutation strategy for different individuals, which achieves higher accuracy than other ADDE variants without adaptive strategy control. The better performance of ADDE is due to make full use of feedback information from both evolution states and individuals. It can achieve a better balance between exploration and exploitation. As a result, the strategy adaptation selection in our algorithm is useful.

2) Individual Parameter F and CR Adaptation Control: Although strategy adaptation selection performs better than other ADDE variants without strategy control, the performance is still sensitive to the individual parameter settings. To research the effect of the individual parameter adaptation control in ADDE, here, three ADDE variants with different F and CR are compared with ADDE on f_1–f_{30}. The F in all ADDE variants is set as 0.5 and CR is set as 0.1, 0.5, and 0.9, respectively. We denote the ADDE variants with CR $=$ 0.1, 0.5, and 0.9 as ADDE(0.1), ADDE(0.5), and ADDE(0.9), respectively. Moreover, we also compare ADDE with its variants with the existing parameter adaptation mechanisms. Herein, we choose two famous parameter adaptation mechanisms used in jDE [44] and JADE [46]. We called the ADDE variants with the parameter adaptation mechanisms used in jDE and JADE are ADDE-jDE and ADDE-JADE, respectively. The detail comparison results are listed in Table S.VII in the supplementary material.

As we can see, CR $=$ 0.1 is able to maintain population diversity and performs better on multimodal functions f_{10}-f_{12}, but it slows the convergence speed and performs worse on the other functions, such as on unimodal function f_1. In contrast, CR $=$ 0.9 achieves fast convergence and obtains more accurate results on the unimodal function, such as f_1, but it may suffer from premature convergence and poor population diversity on multimodal functions on f_{10}–f_{12}. Besides, ADDE-jDE performs even worse than the ADDE variant with a fixed parameter. That may because the adaptive parameter mechanism used in the jDE algorithm is not suitable to the two mutation strategies DE/current-to-rand/1 and DE/current-to-$p$best/1 used in ADDE. It may suitable to the mutation strategy DE/rand/1 used in the jDE algorithm. However, ADDE-JADE performs much better than the ADDE variant with a fixed parameter. That is because in the JADE algorithm, it also used the mutation strategy DE/current-to-$p$best/1 and its adaptive parameter mechanism is suitable to the mutation strategy DE/current-to-$p$best/1. But ADDE-JADE performs still much worse than ADDE since the adaptive parameter mechanism used in ADDE is an improvement of JADE, which can further speed up the individual parameter update process and obtain more successful F and CR. Moreover, the performance of ADDE is independent on the optimization problems. Not only it can obtain more accurate solutions and faster convergence speed on the unimodal functions but also able to keep the population diversity and prevent premature convergence on other complex functions. All in all, from Table S.VII in the supplementary material, we can see that ADDE performs better than ADDE(0.1), ADDE(0.5), ADDE(0.9), ADDE-jDE, and ADDE-JADE on 21, 24, 20, 25, and 18 functions, respectively. In other words, our parameter adaptation control makes ADDE achieve fast convergence and good diversity at the same time.

Next, we further analyze the evolution behavior of CR value in ADDE. From the above experiments, we find that ADDE(0.9) performs well on f_1, f_4, and f_{17}, while ADDE(0.1) is able to obtain better results on f_{10}–f_{12}. Thus, we take these six functions as the representative instances to further verify the effectiveness of parameter adaptation. The variation of CR values in the evolutionary process is observed to check the parameter adaptation mechanism in ADDE.

Since the CR value is controlled by the adaptive parameter μCR, which evolves as the algorithm process in ADDE,

we plot the variation of μCR values on these six functions as the algorithm process in Fig. S2 in the supplementary material for saving space. As we can see, in f_1, f_4, and f_{17} shown in Fig. S2(a)–(c), μCR keeps increasing during the evolutionary process, as we expected. On the contrary, the μCR keeps decreasing during evolution in f_{10}–f_{12} shown in Fig. S2(d)–(f) in the supplementary material, which is consistent to the actual requirement due to the good performance of ADDE(0.1). Therefore, we can say that our parameter adaptation method in the ADDE algorithm is effective and adaptive to the evolutionary states of different problems.

3) Population Parameter N Adaptation Control: Population size N also has a significant influence on the performance of the algorithm. As we mentioned in Section III-D, we wish to use small population size to accelerate the speed of convergence, while using a large population size to improve the population diversity and avoid being trapped in the local optima. Therefore, in order to analyze the role of population size adaptation in ADDE, we compare the performance of ADDE with four ADDE variants with fixed population size setting $N = 100$, 200, 300, and 400, which are defined as ADDE(100), ADDE(200), ADDE(300), and ADDE(400), respectively. The averaged results over 30 runs in each problem are presented in Table S.VIII in the supplementary material in detail.

ADDE(400) maintains a large number of individuals to keep diversity, and its effectiveness can be verified from hybrid functions f_{17} and f_{18}. However, it also slows the speed of convergence on some test functions, such as f_{10}–f_{12}. In contrast, ADDE(100) keeps a relatively small number of individuals, which can accelerate convergence, seen from f_{10}–f_{12}, f_{15}, and f_{16}. However, when dealing with other complex problems such as f_{27}–f_{30}, they may lead to premature convergence. Nevertheless, the method of population size adaptation in ADDE is independent on the optimization problems, it can find a proper population size to achieve a balance between maintaining the population diversity and speeding up the convergence.

In a word, from the Table S.VIII in the supplementary material, ADDE performs better than ADDE(100), ADDE(200), ADDE(300), and ADDE(400) on 22, 20, 19, and 21 functions, respectively. Therefore, we can say that our population size adaptation method is helpful for both keeping diversity and speeding up the convergence.

V. Conclusion

In this article, an ADDE is proposed, which develops a novel master–slave multipopulation DDE algorithm. In ADDE, three populations called exploration population, exploitation population, and balance population are co-evolved concurrently to maximize their strengths by cooperation and enhance the global optimality of DDE. Moreover, different populations adaptively choose different mutation strategies based on the ESE to make full use of the feedback information from both individuals and the whole corresponding population. Besides, we proposed the adaptive parameters (F, CR, and N) method according to HSE and BSI. As a result, ADDE can find a well balance between fast convergence and diversity maintaining.

To evaluate the performance of the proposed algorithm ADDE, we have tested ADDE on a suite of benchmark functions from the CEC 2014 and a suite of real-world application problems from the CEC 2011 test suite. The experimental results show that ADDE can outperform the compared state-of-the-art DDEs and ADEs, on most of the functions based on Wilcoxon's rank-sum test. ADDE exhibits the appropriate exploration and exploitation behaviors which are independent on problems. Therefore, ADDE can quickly adapt to complex and unknown optimization environments and can maintain capabilities of exploration for diversity and exploitation for convergence during the entire evolutionary process.

For future work, we wish to extend the ADDE algorithm to solve more complicated optimization problems such as multiobjective and dynamic optimization problems and apply ADDE to some more real-world problems, such as power systems or cloud computing resources scheduling.

References

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

[2] X. Qiu, J.-X. Xu, Y. Xu, and K. C. Tan, "A new differential evolution algorithm for minimax optimization in robust design," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1355–1368, May 2018.

[3] R. Poláková, J. Tvrdík, and P. Bujok, "Differential evolution with adaptive mechanism of population size according to current population diversity," *Swarm Evol. Comput.*, Apr. 2019. doi: 10.1016/j.swevo.2019.03.014.

[4] X.-G. Zhou and G.-J. Zhang, "Abstract convex underestimation assisted multistage differential evolution," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2730–2741, Sep. 2017.

[5] P. Bujok, J. Tvrdík, and R. Poláková, "Adaptive differential evolution vs. nature-inspired algorithms: An experimental comparison," in *Proc. IEEE Symp. Comput. Intell.*, 2017, pp. 1–8.

[6] Z.-J. Wang *et al.*, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2910721.

[7] X. Qiu, J.-X. Xu, K. C. Tan, and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 232–244, Apr. 2016.

[8] J.-H. Zhong, M. Shen, J. Zhang, H. S.-H. Chung, Y.-H. Shi, and Y. Li, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 512–527, Aug. 2013.

[9] K.-C. Wong, S. K. Yan, Q. Z. Lin, X. T. Li, and C. B. Peng, "Deleterious non-synonymous single nucleotide polymorphism predictions on human transcription factors," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, to be published. doi: 10.1109/TCBB.2018.2882548.

[10] X. T. Li and K.-C. Wong, "Elucidating genome-wide protein-RNA interactions using differential evolution," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 1, pp. 272–282, Jan./Feb. 2019.

[11] J. Sun, S. C. Gao, H. W. Dai, J. J. Cheng, M. C. Zhou, and J. H. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2018.2868493.

[12] N. M. Hamza, D. L. Essam, and R. A. Sarker, "Constraint consensus mutation-based differential evolution for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 447–459, Jun. 2016.

[13] P. Bujok, J. Tvrdík, and R. Poláková, "Comparison of nature-inspired population-based algorithms on continuous optimisation problems," *Swarm Evol. Comput.*, Jan. 2019, Art. no. 100490. doi: 10.1016/j.swevo.2019.01.006.

[14] Z.-J. Wang, Z.-H. Zhan, and J. Zhang, "Distributed minimum spanning tree differential evolution for multimodal optimization problems," *Soft Comput.*, pp. 1–11, Mar. 2019. doi: 10.1007/s00500-019-03875-x.

[15] H. Zhao *et al.*, "Local binary pattern-based adaptive differential evolution for multimodal optimization problems," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2927780.

[16] X.-W. Luo, Z.-J. Wang, R.-C. Guan, Z.-H. Zhan, and Y. Gao, "A distributed multiple populations framework for evolutionary algorithm in solving dynamic optimization problems," *IEEE Access*, vol. 7, pp. 44372–44390, 2019.

[17] X.-F. Liu *et al.*, "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2019.2920887.

[18] Z.-J. Wang, Z.-H. Zhan, and J. Zhang, "Solving the energy efficient coverage problem in wireless sensor networks: A distributed genetic algorithm approach with hierarchical fitness evaluation," *Energies*, vol. 11, no. 12, pp. 1–14, 2018.

[19] Y.-J. Gong *et al.*, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, Sep. 2015.

[20] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2933499.

[21] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 2023–2029.

[22] J. Apolloni, J. García-Nieto, E. Alba, and G. Leguizamón, "Empirical evaluation of distributed differential evolution on standard benchmarks," *Appl. Math. Comput.*, vol. 236, pp. 351–366, Jun. 2014.

[23] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[24] Y.-F. Ge *et al.*, "Distributed differential evolution based on adaptive mergence and split for large-scale optimization," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2166–2180, Jul. 2018.

[25] I. D. Falco, U. Scafuri, E. Tarantino, and A. D. Cioppa, "An asynchronous adaptive multi-population model for distributed differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 5010–5017.

[26] X.-F. Liu *et al.*, "Historical and heuristic-based adaptive differential evolution," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2855155.

[27] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[28] X. Qiu, K. C. Tan, and J.-X. Xu, "Multiple exponential recombination for differential evolution," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 995–1006, Apr. 2017.

[29] 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.

[30] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, W.-N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.

[31] 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.

[32] 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.

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

[34] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, J. Zhang, Y. Li, and Q. Li, "Fast micro-differential evolution for topological active net optimization," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1411–1423, Jun. 2016.

[35] M. Depolli, R. Trobec, and B. Filipiè, "Asynchronous master–slave parallelization of differential evolution for multi-objective optimization," *Evol. Comput.*, vol. 21, no. 2, pp. 261–291, May 2013.

[36] W. Kwedlo and K. Bandurski, "A parallel differential evolution algorithm," in *Proc. IEEE Int. Symp. Parallel Comput. Elect. Eng.*, 2006, pp. 319–324.

[37] K. N. Kozlov and A. M. Samsonov, "New migration scheme for parallel differential evolution," in *Proc. Int. Conf. Bioinformat. Genome Regulat. Structure*, 2006, pp. 141–144.

[38] Y.-F. Ge, W.-J. Yu, J.-J. Li, Z.-W. Yu, and J. Zhang, "Enhancing distributed differential evolution with a space-driven topology," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 4090–4095.

[39] J. Cheng, G. Zhang, and F. Neri, "Enhancing distributed differential evolution with multicultural migration for global numerical optimization," *Inf. Sci.*, vol. 247, pp. 72–93, Oct. 2013.

[40] Y. Xu, Z. Y. Dong, F. J. Luo, R. Zhang, and K. P. Wong, "Parallel-differential evolution approach for optimal event-driven load shedding against voltage collapse in power systems," *IET Gen. Transm. Distrib.*, vol. 8, no. 4, pp. 651–660, Apr. 2014.

[41] A. Glotić, A. Glotić, P. Kitak, J. Pihler, and I. Tičar, "Parallel self-adaptive differential evolution algorithm for solving short-term hydro scheduling problem," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2347–2358, Sep. 2014.

[42] I. De Falco, A. D. Cioppa, D. Maisto, U. Scafuri, and E. Tarantino, "Satellite image registration by distributed differential evolution," in *Applications of Evolutionary Computing*. Berlin, Germany: Springer, 2007, pp. 251–260.

[43] D. R. Penas, J. R. Banga, P. González, and R. Doallo, "A parallel differential evolution algorithm for parameter estimation in dynamic models of biological systems," in *Proc. Int. Conf. Practical Appl. Comput. Biol. Bioinformat.*, 2014, pp. 173–181.

[44] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[45] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 689–707, Oct. 2014.

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

[47] Y.-Z. Zhou, W.-C. Yi, L. Gao, and X.-Y. Li, "Adaptive differential evolution with sorting crossover rate for continuous optimization problems," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2742–2753, Sep. 2017.

[48] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol Comput.*, 2013, pp. 71–78.

[49] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 1658–1665.

[50] J. Brest, M. S. Mauèec, and B. Bošković, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 1188–1195.

[51] J. Brest, M. S. Mauèec, and B. Bošković, "Single objective real-parameter optimization: Algorithm JSO," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 1311–1318.

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

[53] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, 2011.

[54] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.

[55] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[56] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start JADE with knowledge transfer for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1889–1895.

[57] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and School EEE, Nanyang Technol. Univ., Singapore, Rep. 201311, Dec. 2013.

[58] 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.

[59] 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. Eng., Jadavpur Univ., Kolkata, India, and Nanyang Technol. Univ., Singapore, Rep., 2010. [Online]. Available: https://al-roomi.org/multimedia/CEC_Database/CEC2011/CEC2011_TechnicalReport.pdf

[60] A. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proc. Aust. Conf. Artif. Intell.*, 2004, pp. 861–872.

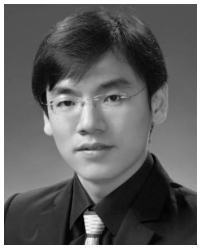
Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees in computer science from the Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, and in environments of cloud computing and big data.

Prof. Zhan was a recipient of the China Computer Federation Outstanding Ph.D. Dissertation Award for his Doctoral dissertation and the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation, the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018, and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of *Neurocomputing* and the *International Journal of Swarm Intelligence Research*.

Zi-Jia Wang (S'15) received the B.S. degree in automation from Sun Yat-sen University, Guangzhou, China, in 2015, where he is currently pursuing the Ph.D. degree.

His current research interests include evolutionary computation algorithms, such as differential evolution, particle swarm optimization, and their applications in design and optimization, such as cloud computing resources scheduling.



Hu Jin (S'07–M'12–SM'18) received the B.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2006 and 2011, respectively.

From 2011 to 2013, he was a Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada. From 2013 to 2014, he was a Research Professor with Gyeongsang National University, Tongyeong, South Korea. Since 2014, he has been with the Division of Electrical Engineering, Hanyang University, Ansan, South Korea, where he is currently an Associate Professor. His current research interests include wireless communications, Internet of Things, and machine learning.

Jun Zhang (F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Professor with Hanyang University, Ansan, South Korea. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, 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 CYBERNETICS, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.