This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Differential Evolution with Auto-Enhanced Population Diversity

Ming Yang, Changhe Li, *Member, IEEE*, Zhihua Cai, and Jing Guan

***Abstract*—In differential evolution (DE) studies, there are many parameter adaptation methods, aiming at tuning the mutation factor F and the crossover probability CR. However, these methods still cannot resolve the issues of population premature convergence and population stagnation. To address these issues, in this paper, we investigate the population adaptation regarding population diversity at the dimensional level and propose a mechanism named auto-enhanced population diversity (AEPD) to automatically enhance population diversity. AEPD is able to identify the moments when a population becomes converging or stagnating by measuring the distribution of the population in each dimension. When convergence or stagnation is identified at a dimension, the population is diversified at that dimension to an appropriate level or to eliminate the stagnation issue. The AEPD mechanism was incorporated into a popular DE algorithm and it was tested on a set of 25 CEC2005 benchmark functions. The results showed that AEPD significantly improved the performance of the original algorithms. In addition, AEPD helped the algorithms become less sensitive to population size, a parameter widely considered problem dependent for many DE algorithms. The DE algorithm with AEPD also has a superior performance in comparison with several other peer algorithms.**

***Index Terms*—Differential evolution, population adaptation, population diversity auto-enhancement.**

I. INTRODUCTION

DIFFERENTIAL evolution (DE), introduced by Storn and Price [1], is a simple yet powerful evolutionary algorithm (EA) for global optimization problems. Nowadays DE has become one of the most frequently used EAs for solving global optimization problems [2], mainly because it has good convergence property and is principally easy to understand. Its effectiveness and efficiency have been successfully demonstrated in many real-world applications, such as procedural modeling, engineering design optimization, power flow optimization, economic and environmental dispatch optimization, and so on. Similar to other types of EA, DE also simulates the natural evolution process via mutation, recombination and selection to evolve a population of initially random solutions into an optimal solution.

What distinguish DE from other EAs are its two components, which significantly influence DEs performance: the trial vector generation strategies (mutation and crossover operators) and the control parameters. There are many different trial vector generation strategies for DE, each of which seems to be suitable for some particular tasks or for solving a certain type of problems [3], [4]. There are three control parameters in DE: the amplification factor of the difference vector—F, the crossover control parameter—CR, and the population size—NP. All these three parameters significantly affect the performance of DE algorithms [5]–[7]. The optimal choice of the three control parameters in DE often depends on the problems to be solved. For a specific problem, one may need to spend a huge amount of time to try and fine-tune the corresponding parameters. To address this issue, several adaptive and self-adaptive DE algorithms regarding F and CR were developed to solve general problems efficiently [8]–[10].

Two other issues significantly affecting the performance of DE are stagnation and premature convergence [11]. Stagnation is the situation where DE is unable to generate a better child solution for ever but the population has not converged [12]. That is, even though the population diversity is not poor, the algorithm is still unable to find any better solution. Premature convergence is the situation, where the population converges to local optima, due to the loss of diversity.

The problem of stagnation and premature convergence might become worse when some adaptive controls for the population size NP are introduced. Current adaptive methods to control NP [13], [14] share one common problem: increasing the population size decreases the possibility of finding correct search directions while decreasing population size increases the possibility of premature convergence and stagnation. The optimal value of NP also seems to be problem-dependent. A guideline in [1] suggested that a reasonable value for NP could be between five and ten times the dimensionality of a problem. However, a study in [15] argued that a population size lower than problem dimensionality can be optimal in many cases.

Another issue is to find the effective movements for individuals in the population. Of all the possible movements by a population, some movements are beneficial in the search for the global optimum while some others are ineffective and result in a waste of computational effort [3]. How to reduce the ineffective moves is an open issue for DE algorithms.

Manuscript received December 16, 2013; revised April 23, 2014 and July 5, 2014; accepted July 8, 2014. This work was supported in part by the fund of the National Natural Science Foundation of China under Grant 61305086, Grant 61203306 and Grant 61305079, in part by the Fundamental Research Funds for the Central Universities under Grant CUG120114, and in part by the Key Project of the Natural Science Foundation of Hubei Province, China, under Grant 2013CFA004. This paper was recommended by Associate Editor Y. Jin. *(Corresponding author: Changhe Li.)*

M. Yang, C. Li, and Z. Cai are with the School of Computer Science, China University of Geosciences, Wuhan 430074, China (e-mail: yangming0702@gmail.com; changhe.lw@gmail.com; zhcai@cug.edu.cn).

J. Guan is with the China Ship Development and Design Center, Wuhan 430064, China (e-mail: g_jing0414@163.com).

Digital Object Identifier 10.1109/TCYB.2014.2339495

To address the above issue of reducing the ineffective moves and to solve the problems of population premature and stagnation, we propose a novel idea to improve the performance of a general DE variant by diversifying a small size of population at the dimensional level.

The rest of the paper is organized as below. Section II reviews the related DE algorithms. The motivation of this paper is illustrated in Section III. The mechanism of the auto-enhanced population diversity is introduced in Section IV. Section V presents experimental results. Finally, conclusions are given in Section VI.

II. Related Work

A. Related DE Algorithms

Many modifications and extensions have been made on DE research. Fan and Lampinen [16] proposed a new mutation scheme, trigonometric mutation operator, for DE to enhance its performance. This modification enables the algorithm to get a better trade-off between the convergence rate and robustness. Rahnamayan *et al.* [17] recently proposed an opposition-based differential evolution (ODE) for faster global search and optimization. The conventional DE was enhanced by utilizing the opposition number based optimization concept in two levels, namely, population initialization and generation jumping. In order to achieve a better balance between explorative and exploitative tendencies, Das *et al.* [18] proposed two kinds of topological neighborhood models and employed them into the mutation of DE. A compact differential evolution (cDE) algorithm was proposed in [19], where it does not process a population of solutions but evolves in a similar way to the original DE algorithm [1]. Mallipeddi and Suganthan [20] proposed a DE (EPSDE) with an ensemble of mutation and crossover strategies with associated control parameters. In EPSDE, a pool of distinct mutation and crossover strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offsprings. A similar composite DE algorithm (CoDE) was proposed in [21]. CoDE uses three trial vector generation strategies and three control parameter settings, and it randomly combines them to generate trial vectors. In [22], a proximity-based mutation operator was proposed in an attempt to efficiently guide the evolution of the population toward the global optimum. This operator modifies the random selection of parent individuals during mutation by assigning each individual a probability of selection, which is inversely proportional to the distance between the parent and the mutated individual. Dorronsoro and Bouvry [23] designed a heterogeneous distributed algorithm (HdDE) with two islands. One island employs the classical DE/rand/1/bin algorithm and the second one uses a mutation strategy, called (GPBX-α). In [24], DE was reviewed under unified framework, and functional requirements of initialization, selection, generation and replacement are evaluated. This study outlines a direction for advancing evolutionary computation methods. A clear and fundamental algorithmic linking was established between particle swarm optimization algorithm and genetic algorithms in [25]. For an attempt to design efficient optimization algorithms, the concept of algorithmic linking suggests more efforts should be made in establishing equivalence between different genetic, evolutionary and other nature-inspired or non-traditional algorithms.

Over the past few years researchers have investigated different ways of improving the performance of DE algorithms by tuning the control parameters. Liu and Lampinen [26] introduced a fuzzy adaptive differential evolution (FADE) using fuzzy logic controllers, whose inputs incorporate the relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operation. Qin and Suganthan [27] proposed a self-adaptive differential evolution (SaDE), where the choice of learning strategy and the two control parameters F and CR do not require predefining. During evolution, suitable learning strategy and parameter settings are gradually self-adapted, according to the learning experience. Brest *et al.* [28] proposed a self-adaptation scheme for the DE control parameters. They encoded the control parameters F and CR into individuals and adjusted them with two probabilities. In their algorithm, called jDE, a set of F and CR values was assigned to each individual in the population. The better values of these encoded control parameters lead to better individuals, which, in turn, are more likely to survive and produce offspring. jDE was further extended by adapting two mutation strategies and the new algorithm was named jDE-2 [29]. In order to avoid the need for problem specific parameter tuning, an adaptive DE-variant, called JADE, was proposed [30]. The algorithm implements a new mutation strategy, named DE/current-to-*p*best, and uses an optional external archive to track the previous history of success and failures. The algorithm updates the control parameters F and CR, associated with each individual in the population, based on their historical record of success. A generalized adaptive DE algorithm (GaDE) [31] is also based on the record of success.

B. Diversity Measurements and Diversity Regain

Population diversity has an important influence on the performance of evolutionary algorithms. Two kinds of approaches have been proposed for evaluating the genotypic diversity. The first is based on a measurement of the distance between individuals. This distance may be evaluated by the mean spatial position of the population [32] to the position of the fittest individual [33], or the position of each individuals. Measures of the latter type include the pairwise measure [34]–[36] and the maximum distance between two individuals [35]. The Euclidian distance is common for distance estimation with real-coded genes. The second kind of approaches scans gene frequency. Gouvêa and Araújo [37] used a representative gene to characterize population diversity. Instead of measuring diversity on dimensions as usual, only one representative gene was selected for diversity measurement. This approach requires that the chosen gene must be representative. Therefore, to avoid a misleading diversity estimation, an evaluation obtained from the diversity measure of all genes was proposed in [38]. This kind of method was also used in [39] to determine the population diversity. Morrison and De Jong [40] proposed the moment of inertia, which was calculated based on the concept of the moment-of-inertia in physics, to measure population diversity. Corriveau *et al.* [41] surveyed the genotypic diversity measures published over the years for real-coded representations, and compared them based on a new benchmark.

For DE, several methods have been proposed to maintain population diversity. The first kind of these methods changes the values of parameters F and CR in order to increase the variance of the trial vectors population. The adaptive DE algorithms such as FADE [26], SaDE [27], jDE [28] and JADE [30], introduced in Section II-A, belong to this kind. Besides these algorithms, Zaharie [42] theoretically analyzed the relationship between the control parameters and population variance and proposed a strategy to determine the excepted values of F, CR, and NP [43], which can control the decreasing rate of population variance. The second kind of methods perturbs the population elements by reinitializing some individuals [32], [44] to maintain population diversity for solving dynamic optimization. There is also a selectively destructive strategy [45], in which a percentage of the converged genes remains unchanged to begin the next convergence stage, while the other genes are reinitialized. DE can employ other algorithms or mutation operators as a mutation operator to increase the population diversity, like the algorithmic linking in [25], the cooperative coevolving algorithm in [46] and the adaptive DE algorithm in [47]. The third kind of methods adopts structured populations with controlled migration to main population diversity. The DynDE algorithm tried to keep each population on a different peak by the exclusion strategy [48]. The HdDE [23] algorithm adopted two subpopulations, and each subpopulation employs a different mutation strategy. The two subpopulations exchange information at several certain generation points. The above three kinds of methods except the second kind are not able to address the issue that the algorithm stops evolving when the population converges. Most methods of the second kind are not able to detect accurately the moment when a population needs to be diversified. These methods would also rediversify the population even though the population is not converged. This slows down the convergence of DE algorithm and may deteriorate the search.

III. Population Diversity at the Dimensional Level

DE employs the mutation operation to produce a mutant vector $\mathbf{v}_{i,G}$ with respect to an individual $\mathbf{x}_{i,G}$, so-called target vector, at each generation G. For each D-dimensional target vector $\mathbf{x}_{i,G}$, its associated mutant vector $\mathbf{v}_{i,G} = (v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G})$ can be generated via a certain mutation strategy. The four most frequently used mutation strategies in DE are listed as follows.

1) DE/rand/1

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}). \tag{1}$$

2) DE/best/1

$$\mathbf{v}_{i,G} = \mathbf{x}_{\text{best},G} + F \cdot (\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G}). \tag{2}$$

3) DE/current-to-best/1

$$\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + K \cdot (\mathbf{x}_{\text{best},G} - \mathbf{x}_{i,G}) + F \cdot (\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G}). \tag{3}$$

4) DE/rand-to-best/1

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + K \cdot (\mathbf{x}_{\text{best},G} - \mathbf{x}_{r_1,G}) + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}). \tag{4}$$

The indices r_1, r_2, r_3 are mutually exclusive integers. They are randomly generated within the range $[1, NP]$, which are also different from the index i. These indices are randomly generated once for each mutant vector. The scaling factors F and K are positive real numbers for scaling the difference vector, and usually $K = F$. $\mathbf{x}_{\text{best},G}$ is the individual with the best fitness value at the generation G.

When a population gets trapped in a local optimum, individuals are almost the same. The difference vectors generated by the mutation strategies in DE are almost zero. Therefore, the mutation strategies cannot generate new vectors that are far beyond the location of the current population. As a result, the population cannot jump out of the local optimum. This is a fatal defect of DE. Take the 5-dimensional sphere function for example. It is a classical unimodal function, which is described as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2, x_i \in [-100, 100] \tag{5}$$

where $D = 5$ and the objective value of the global optimum is 0 at $\mathbf{x} = (0, \ldots, 0)$. Fig. 1 shows locations changes for five individuals with the DE/rand/1/bin algorithm. After 200 function evaluations, all the individuals in the population are almost the same in each dimension: $x_1 \approx -4.2$, $x_2 \approx -32.16$, $x_3 \approx 8.731$, $x_4 \approx -13.34$, and $x_5 \approx 0.1813$ [see Fig. 1(a)–(e)]. The population diversity is very small, and the population has been trapped at the location described above. Even if the evolution goes on, the population will no longer improve. This issue may be much more serious for multimodal problems than unimodal problems. Fig. 1(f) shows the changes of standard deviation (std) in each dimension for all the individuals. In this figure, the standard deviation value in dimension 1 (variable x_1) decreases to almost zero at the end of the run (i.e., $\text{std}_1 \approx 10^{-15}$). However, in the other dimensions, the standard deviation values of individual solutions at the end of the run are far from zero compared with the standard deviation of x_1 (i.e., $\text{std}_i \approx 10^{-4}$, $i = 2,\ldots,5$), which indicates that the population can be further improved in these dimensions.

The preliminary experimental results on the sphere function suggest that the evolutionary progress in each dimension is asynchronous with each other. Although several measurements were proposed to evaluate population diversity (see Section II-B), they do not consider the difference among dimensions. The diversity could be small in some dimensions and not so small in some others. Therefore, it would be better to measure diversity separately in each dimension, and the information about which dimension has good diversity should be kept in each individual.

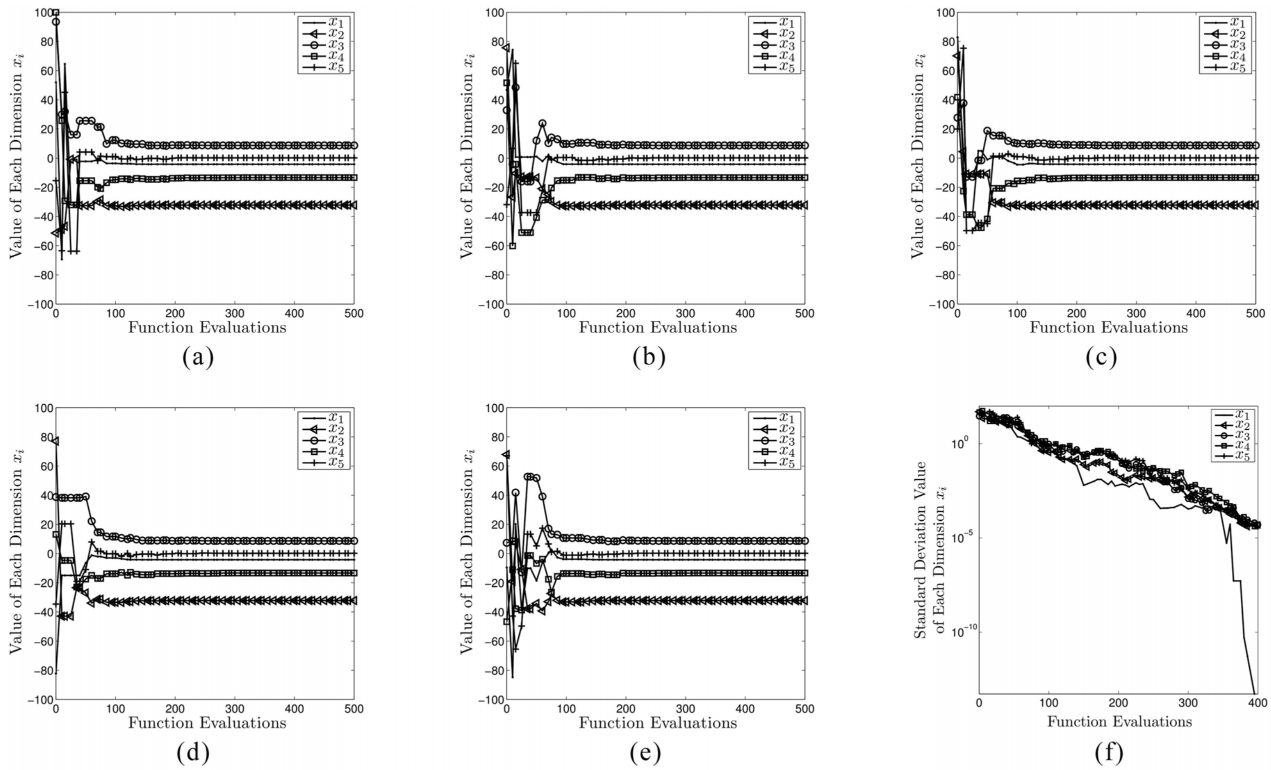

Fig. 1. Changes at the dimensional level of each individual in the DE/rand/1/bin algorithm on the sphere function in five dimensions with NP = 5 in a typical run. (a) First individual. (b) Second individual. (c) Third individual. (d) Fourth individual. (e) Fifth individual. (f) Standard deviation values of each dimension.

IV. Auto-Enhanced Population Diversity

The population should be rediversified only when it is converging or stagnant. Before this rediversification, it is essential to confirm the population is converging or stagnant. Population premature convergence is the situation, where the population converges to a local optimum, due to the loss of diversity. Population stagnation is the situation where the algorithm is unable to generate a better child solution for ever but the population has not converged. The methods of identifying the two situations should be different. And regarding the fact that the evolutionary progress in each dimension is asynchronous with each other (see Section III), the two situations should be identified at the dimensional level.

To identify the dimension that needs to be diversified when the population converges or stagnates, a new diversity enhance mechanism named auto-enhanced population diversity (AEPD) is introduced in this section.

A. Identification of Population Convergence

In order to measure population diversity in each dimension, the mean and the standard deviation of individuals' variables in the jth dimension at the Gth generation is calculated by

$$m_{j,G} = \frac{1}{NP}\sum_{i=1}^{NP} x_{i,j,G} \tag{6}$$

$$\mathrm{std}_{j,G} = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}(x_{i,j,G} - m_{j,G})^2} \tag{7}$$

where $m_{j,G}$ and $\mathrm{std}_{j,G}$ are the mean and standard deviation of the population in the jth dimension at generation G, respectively. The value of $\mathrm{std}_{j,G}$ is used to measure population diversity in the jth dimension.

A smaller value of $\mathrm{std}_{j,G}$ indicates a lower population diversity in the jth dimension. Theoretically, when $\mathrm{std}_{j,G}$ is equal to zero, the population has completely converged or completely lost its diversity in the jth dimension. For most EAs including the DE algorithm, population diversity is large at the beginning of the evolutionary process and the value of $\mathrm{std}_{j,G}$ is large as well. Experimental studies show that it takes a very long time for $\mathrm{std}_{j,G}$ to decrease to zero. Therefore, practically, we use a small value $\omega_{j,G}$ instead of zero to identify the moments when population diversity becomes small [see (8a)]. $\bar{r}_{j,G}$ is a flag to denote whether the population has converged in the jth dimension at the Gth generation

$$\bar{r}_{j,G} = \begin{cases} 1 & \text{if } \mathrm{std}_{j,G} \le \omega_{j,G} \quad \text{(8a)} \\ 0 & \text{otherwise.} \quad \text{(8b)} \end{cases}$$

If $\mathrm{std}_{j,G}$ is not greater than $\omega_{j,G}$, $\bar{r}_{j,G}$ is set to 1 to indicate that the population has converged in the jth dimension. The value of $\omega_{j,G}$ is defined as follows:

$$\omega_{j,G} = \min(T, \theta_{j,G}) \tag{9}$$

where

$$\theta_{j,G} = \begin{cases} |m_{j,G} - MR_j| \cdot T & \text{if } \mathrm{std}_{j,G} \le T \quad \text{(10a)} \\ T & \text{otherwise} \quad \text{(10b)} \end{cases}$$

and $T = 10^{-3}$, MR_j is the value of $m_{j,G}$ just before the last diversity enhancement operation for the jth dimension (the initial value of MR_j is set to the mean value $m_{j,G}$ of the initialized population).

Generally speaking, the population will gradually converge and accordingly the value of $\text{std}_{j,G}$ will get smaller. When $\text{std}_{j,G} \leq T$, the population is assumed to be converged in the jth dimension. Then, the value of $|m_{j,G} - MR_j|$ is taken into consideration. If $|m_{j,G} - MR_j|$ is small, the population is likely to converge to the same location as it did the last time it converged. In such case, $\omega_{j,G}$ is set to $|m_{j,G} - MR_j| \cdot T$ [see (10a) and (9)], a smaller value than T. This can ensure that the population would only be diversified until it converges enough to the optimum. On the contrary, if $|m_{j,G} - MR_j|$ is large, the population is likely to converge to a different optimum from the last time. In such case, $\omega_{j,G}$ is set to T [see (9)] to make sure that the population is diversified in a short time after it starts converging.

The value of $\omega_{j,G}$ is able to adapt to the changing information of the population during the evolutionary process. Because of this mechanism, the population is able to both sufficiently exploit a revisited optimum for unimodal problems (e.g., the sphere function) and efficiently explore new promising optima for multimodal problems.

B. *Identification of Population Stagnation*

In the case of stagnation, the distribution of population does not change for ever. In practice if the distribution, the mean and the standard deviation of population, has not changed for several successive generations, we consider the population stagnant.

Suppose that $\lambda_{j,G}$ denotes the number of successive generations where the values of $m_{j,G}$ and $\text{std}_{j,G}$ remain unchanged. For the jth dimension at the Gth generation, the value of $\lambda_{j,G}$ is calculated by

$$\lambda_{j,G} = \begin{cases} \lambda_{j,G-1} + 1 & \text{if } m_{j,G} = m_{j,G-1} \text{ and} \\ & \text{std}_{j,G} = \text{std}_{j,G-1} \quad \text{(11a)} \\ 0 & \text{otherwise} \quad \text{(11b)} \end{cases}$$

and $\lambda_{j,0} = 0$. $\hat{r}_{j,G}$ is a flag to denote whether the population has stagnated in the jth dimension at the Gth generation

$$\hat{r}_{j,G} = \begin{cases} 1 & \text{if } \lambda_{j,G} \geq UN \quad \text{(12a)} \\ 0 & \text{otherwise} \quad \text{(12b)} \end{cases}$$

where UN is an integer, whose value equals NP (Our experimental results show that the larger population size, the more generations it will take for the population to enter a stable stagnation state. Therefore, we use $UN = NP$). If $\lambda_{j,G} \geq UN$, $\hat{r}_{j,G}$ is set to 1 to indicate that the population is likely to start stagnating in the jth dimension as its distribution has not changed for several successive generations.

C. *Enhancement of Population Diversity*

$r_{j,G}$ is a flag to denote whether the population needs to be rediversified in the jth dimension at the Gth generation

$$r_{j,G} = \begin{cases} 1 & \text{if } \bar{r}_{j,G} = 1 \text{ or } \hat{r}_{j,G} = 1 \quad \text{(13a)} \\ 0 & \text{otherwise.} \quad \text{(13b)} \end{cases}$$

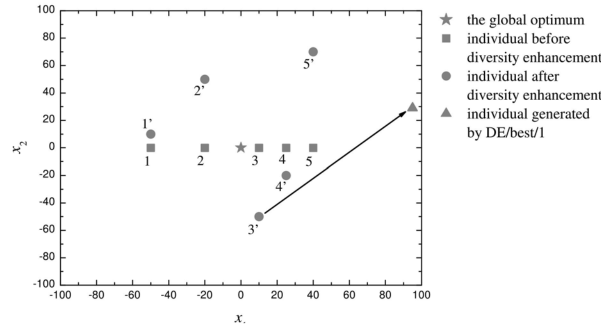

Fig. 2. Population change by the way that the value of each individual at each dimension is regenerated separately for the sphere function in 2-dimensions, when $NP = 5$.

If the population starts converging ($\bar{r}_{j,G} = 1$) or stagnating ($\hat{r}_{j,G} = 1$) in the jth dimension, $r_{j,G}$ is set to 1 to indicate that the population needs to be rediversified in this dimension.

However, our experimental results show that immediate rediversifying a converged or stagnated population in a dimension identified by the above approach may deteriorate the search. Take the 2-D sphere function [see (5)] as an example. Fig. 2 illustrates such a situation where we use a population of five individuals to solve the sphere function. The square and round points are locations of the five individuals before and after a rediversification, respectively. According to (13a), the second dimension (x_2) of all the five individuals will be rediversified when the population converges in that dimension. Before the rediversification, individual 3 is the best. However, after the rediversification, individual 4 becomes the best one. As a result, some individuals generated by the DE/best/1 mutation (e.g., individual 3) are likely to move far away from the global optimum due to the new best individual 4. Although the values in the first dimension (x_1) for all five individuals are not affected by the rediversification in the second dimension, the choice of the best individual is affected. Therefore, the rediversification will disturb the evolution of the population, even in those dimensions that are not selected for diversification. Experimental results show that this affection will become more serious for functions that have some linkages between dimensions. In order to solve this issue, rediversification should be performed only when the condition below is met

$$z_G = \begin{cases} 1 & \text{if } \sum_{j=1}^{D} r_{j,G} = D \text{ or } randu < c \quad \text{(14a)} \\ 0 & \text{otherwise} \quad \text{(14b)} \end{cases}$$

where z_G is a flag to denote whether the population is rediversified at the Gth generation and $randu$ is a uniformly distributed random number within [0, 1]. For high-dimensional problems, it would need a large number of function evaluations to flag all the dimensions for rediversification (i.e., $\sum_{j=1}^{D} r_{j,G} = D$). To alleviate this issue, the population will be also rediversified with a small probability [i.e., c in (14a)] in any dimension j where $r_{j,G} = 1$. We use $c = 10^{-3}$ in this paper. This is because AEPD-JADE performs best with that value (experimental results of the sensitivity analysis of c are provided in Tables VI and VII in the supplementary material listed in the Appendix).

To diversify the population in the jth dimension to handle the above two situations, we regenerate $x_{i,j,G+1}$, the jth value of $\mathbf{x}_{i,G+1} = (x_{i,1,G+1}, x_{i,2,G+1}, \ldots, x_{i,D,G+1})$, $i \in [1, NP]$, as follows:

$$x_{i,j,G+1} = \mathrm{low}_{j,G} + (\mathrm{up}_{j,G} - \mathrm{low}_{j,G}) \cdot randn_{j,G} \tag{15}$$

where

$$\mathrm{low}_{j,G} = \min(m_{j,G}, x_{\mathrm{low},j})\text{[1]} \tag{16}$$

$$\mathrm{up}_{j,G} = \max(m_{j,G}, x_{\mathrm{up},j}).\text{[1]} \tag{17}$$

$x_{\mathrm{low},j}$ and $x_{\mathrm{up},j}$ are predefined lower and upper bounds for the jth dimension, respectively; $randn_{j,G}$ is a random number within the range [0,1] and follows a normal distribution of mean $\mu_{j,G}$ and variance $\sigma_{j,G}^2$. The values of $\mu_{j,G}$ and $\sigma_{j,G}$ are as follows:

$$\mu_{j,G} = \frac{m_{j,G} - \mathrm{low}_{j,G}}{\mathrm{up}_{j,G} - \mathrm{low}_{j,G}} \tag{18}$$

$$\sigma_{j,G} = \exp\left(\frac{-a \cdot k}{D}\right) \cdot \bar{\sigma}_{j,G} \tag{19}$$

where

$$\bar{\sigma}_{j,G} = \max(\mu_{j,G}, 1 - \mu_{j,G}). \tag{20}$$

k is the number of the current function evaluations and $a = 0.0005$ (preliminary experiments on the sensitivity analysis of a show that AEPD-JADE performs best with $a = 0.0005$, and the experimental results are provided in Tables VIII and IX in the supplementary material listed in the Appendix). Equation (19) means that $\sigma_{j,G}$ decreases when k increases, but the rate of decrease becomes slower when k becomes larger. And the decline of $\sigma_{j,G}$ becomes slow as the number of dimensions increases. If the value of $\sigma_{j,G}$ is smaller than 10^{-3}, $\sigma_{j,G}$ is set to 10^{-3}. Equations (18) and (19) mean that, in the next generation G+1 the newly generated variable $x_{i,j,G+1}$ would have a high probability of being generated close to the current position $m_{j,G}$, and a small probability of being generated far away from the current position. In comparison to random reinitialization methods, this mechanism enables algorithms to be able to both further exploit the local area if it has not been sufficiently searched and explore other promising areas.

Algorithm 1 illustrates the framework of the auto-enhanced population diversity (AEPD). The fitness of rediversified individuals are evaluated and these evaluations are counted. In order to prevent the current best solution from being destroyed, the AEPD approach does not rediversify the best solution found so far by the whole population. Note that, if there are multiple copies of the best solution, only one copy is kept. This is because more individuals are kept, more possible the population converges again to the revisited optimum. It is more serious for the mutation strategies in DE utilizing the information of the best individual, such as DE/best, DE/current-to-best and so on.

[1]For unconstrained functions, $\mathrm{low}_{j,G}$ and $\mathrm{up}_{j,G}$ are allowed to be set beyond of the predefined bounds. This is helpful to solve functions whose global optima are outside of predefined range.

Algorithm 1 Auto-Enhanced Population Diversity (AEPD)

1: Compute $m_{j,G}$ and $\mathrm{std}_{j,G}$ for each dimension $j = 1, 2, \ldots, D$;
2: **for** each dimension $j = 1, 2, \ldots, D$ **do**
3:   Compute $r_{j,G}$ using Eq. (13);
4: **end for**
5: Compute z_G using Eq. (14);
6: $flagM = 0$;
7: **if** z_G equals 1 **then**
8:   **for** each dimension $j = 1, 2, \ldots, D$ **do**
9:     **if** $r_{j,G}$ equals 1 **then**
10:       $flagM = 1$;
11:       **for** each individual $\mathbf{x}_{i,G}$, $i = 1, 2, \ldots, NP$ **do**
12:         **if** $\mathbf{x}_{i,G}$ is not the best individual **then**
13:           Regenerate $x_{i,j,G+1}$ of the next generation using Eq. (15);
14:         **end if**
15:       **end for**
16:       $MR_j = m_{j,G}$;
17:     **end if**
18:   **end for**
19:   **if** $flagM$ equals 1 **then**
20:     Evaluate the fitness for each individual $\mathbf{x}_{i,G+1}$, $i = 1, 2, \ldots, NP$, except the best individual;
21:   **end if**
22: **end if**

In Algorithm 1, the computational complexity of $m_{j,G}$ and $\mathrm{std}_{j,G}$ (step 1) is $O(2 \cdot D \cdot NP)$. For computing $r_{j,G}$ (from step 2 to step 4), the computational complexity is $O(D)$. And the computational complexity of z_G (step 5) is $O(D)$. The total computational complexity of measuring population diversity is $O((2 \cdot NP + 2) \cdot D)$. The computational complexity of rediversification (from step 8 to step 18) is $O(D \cdot NP)$. The fitness evaluations (step 20) are not considered, because these are counted into the whole function evaluations. The total computational complexity of AEPD is $O((3 \cdot NP + 2) \cdot D)$. The computational complexity of AEPD is smaller than that of traditional methods of computing pairwise population diversity, whose computational complexity is $O(D \cdot NP^2)$ [41], and is in a same order of magnitude as the method in [38].

AEPD reinitializes individuals to enhance population diversity. It belongs to the second kind of methods for maintaining population diversity (see Section II-B). AEPD has three differences in comparison with existing methods in this category. Firstly, AEPD discriminates between population convergence and population stagnation and identifies the two cases by different approaches. Secondly, considering the fact that the evolutionary progress in different dimension is asynchronous with each other (see the experimental results in Section III), AEPD is able to determine the moment when the population needs rediversification at the dimensional level, i.e., the threshold value of $\omega_{j,G}$ is different for different dimensions. Thirdly, the value of $\omega_{j,G}$ is adaptive according to the changing information of the population during the evolutionary process. Therefore, we believe AEPD would improve

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 2 AEPD-JADE Algorithm

1: Set $\mu_{CR} = 0.5$, $\mu_F = 0.5$ and $A = \emptyset$;
2: Generate uniform random individuals for the initial population P_0;
3: Evaluate the fitness for each individual in P_0;
4: $G = 1$;
5: **while** the stop criterion is not satisfied **do**
6: $S_F = \emptyset$, $S_{CR} = \emptyset$;
7: **for** each individual $\mathbf{x}_{i,G} \in P_G$ **do**
8: Generate $CR_i = randn_i(\mu_{CR}, 0.1), F_i = randc_i(\mu_F, 0.1)$; /**randn*$_i$ is a random number within the range [0,1] and follows a normal distribution. *randc*$_i$ is a random number within the range [0,1] and follows a Cauchy distribution. */
9: Randomly choose $\mathbf{x}^p_{\text{best},G}$ as one of the $100p\%$ best vectors;
10: Randomly choose $\mathbf{x}_{r_1,G} \neq \mathbf{x}_{i,G}$ from the current population P_G;
11: Randomly choose $\tilde{\mathbf{x}}_{r_2,G} \neq \mathbf{x}_{r_1,G} \neq \mathbf{x}_{i,G}$ from $P_G \bigcup A$;
12: $\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + F_i \cdot (\mathbf{x}^p_{\text{best},G} - \mathbf{x}_{i,G}) + F_i \cdot (\mathbf{x}_{r_1,G} - \tilde{\mathbf{x}}_{r_2,G})$;
13: j_{rand} = rndint(1,D);
14: **for** j = 1 to D **do**
15: **if** rnd(0,1)$<$ CR_i or $j == j_{rand}$ **then**
16: $u_{i,j,G} = v_{i,j,G}$;
17: **else**
18: $u_{i,j,G} = x_{i,j,G}$;
19: **end if**
20: **end for**
21: **if** $u_{i,j,G} \notin [x_{\text{low},j}, x_{\text{up},j}]$ **then**
22: Use Eq. (21) to map $u_{i,j,G}$ to be in the search range $[x_{\text{low},j}, x_{\text{up},j}]$;
23: **end if**
24: Evaluate the offspring $\mathbf{u}_{i,G}$;
25: **if** $\mathbf{u}_{i,G}$ is better than $\mathbf{x}_{i,G}$ **then**
26: $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G}$; $\mathbf{x}_{i,G} \rightarrow A$; $CR_i \rightarrow S_{CR}$; $F_i \rightarrow S_F$;
27: **else**
28: $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$;
29: **end if**
30: **end for**
31: Randomly remove solutions from A so that $|A| \leq NP$;
32: $\mu_{CR} = (1-c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR})$;
33: $\mu_F = (1-c) \cdot \mu_F + c \cdot mean_L(S_F)$;
34: *Implement the auto-enhancement of population diversity using Algorithm 1*;
35: $G = G + 1$;
36: **end while**

the performance of traditional DE variants by considering these issues.

D. DE Algorithms With AEPD

It is very simple to apply AEPD to existing DE algorithms. The AEPD mechanism is performed after DE operators at each generation. For consistency, we name a DE algorithm, which uses the AEPD approach, AEPD-DE in this paper. For example, Algorithm 2 presents the pseudo-code of AEPD-JADE, which combines AEPD with the JADE [30] algorithm. Comparing with the original DE algorithms, AEPD-DE algorithms only need one extra step (see step 34 in Algorithm 2).

Note that, for all DE algorithms in this paper, if a trial vector $\mathbf{u}_i$ is generated outside of the search range in the jth dimension after the mutation, we map $u_{i,j}$ to the valid search range as follows:

$$u_{i,j} = \begin{cases} (N \cdot F_{\max} \cdot x_{\text{up},j} - x_{\text{low},j} + u_{i,j})/(N \cdot F_{\max}) \\ \qquad\qquad \text{if } u_{i,j} < x_{\text{low},j} \\ (N \cdot F_{\max} \cdot x_{\text{low},j} - x_{\text{up},j} + u_{i,j})/(N \cdot F_{\max}) \\ \qquad\qquad \text{if } u_{i,j} > x_{\text{up},j} \end{cases} \tag{21}$$

where $F_{\max}$ is the maximal value of the parameter F; N is the number of difference vectors in the mutation operation of DE; $x_{\text{low},j}$ and $x_{\text{upp},j}$ are the predefined lower and upper bounds, respectively.

V. Experimental Study

Twenty five test instances proposed in the CEC 2005 special session on real-parameter optimization were used to study the performance of the proposed AEPD. A detailed description of these test instances can be found in [49]. In these functions, f_1–f_5 are unimodal functions, f_6–f_{12} are basic multimodal functions, f_{13}–f_{14} are expanded multimodal functions, and f_{15}–f_{25} are hybrid composition functions. Note that, functions f_7 and f_{25} have no boundary constraint. Therefore, for these two functions, for all algorithms we do not remap individuals that are generated outside the initial search range during the search.

For each algorithm and each test function, 30 independent runs were conducted with Max*FEs* = 10000 $\times$ D function evaluations as the termination criterion where D is the number of dimensions of the problems. In our experimental studies, the results are averaged over 30 independent runs for measuring the performance of each algorithm.

A. Comparison of AEPD-JADE With Other Algorithms

1) Peer Algorithms: To compare the performance of AEPD-JADE, we select ten other peer algorithms, including eight classic DE variants and two non-DE algorithms. The eight DE variants are DE/rand/1/bin [1], JADE [30], jDE [28], SaDE [27], CoDE [21], Pro DE/rand/1/bin [22], HdDE [23], and EPSDE [20]. The two non-DE algorithms are CLPSO [50] and IPOP-CMA-ES [51]. CLPSO is a comprehensive learning particle swarm optimizer, which showed good performance in solving multimodal problems [50]. IPOP-CMA-ES is a covariance matrix adaptation evolution strategy (CMA-ES) algorithm [52] with mechanisms of restart and increasing population size.

The parameter settings of each peer algorithm are the same as those used in their original papers, except that the Quasi-Newton local search is disabled in SaDE, because the authors did not state how to implement the Quasi–Newton local search in this algorithm [27]. The parameters of DE/rand/1/bin were set to F = 0.5 and CR = 0.9, which were recommended

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
STATISTICAL RESULTS WHEN *NP* IS THE VALUE RECOMMENDED IN THEIR ORIGINAL PAPERS: *b*/*n*/*w*, WHERE *b*, *n*, AND *w* DENOTE THE NUMBER OF FUNCTIONS ON WHICH AEPD-JADE PERFORMS SIGNIFICANTLY BETTER THAN, STATISTICALLY EQUIVALENT TO, AND SIGNIFICANTLY WORSE THAN THEIR PEER ALGORITHMS, RESPECTIVELY

	D=30									
	DE/rand/1/bin	JADE	jDE	SaDE	CoDE	Pro DE/rand/1/bin	HdDE	EPSDE	CLPSO	IPOP-CMA-ES
AEPD-JADE	11/8/6	11/9/5	15/5/5	20/3/2	23/1/1	11/9/5	13/6/6	24/0/1	25/0/0	15/3/6
	D=50									
	DE/rand/1/bin	JADE	jDE	SaDE	CoDE	Pro DE/rand/1/bin	HdDE	EPSDE	CLPSO	IPOP-CMA-ES
AEPD-JADE	16/1/8	14/4/7	15/2/8	16/4/5	21/1/3	16/1/8	14/5/6	20/4/1	25/0/0	13/6/5

TABLE II
STATISTICAL RESULTS WHEN *NP* = 20: *b*/*n*/*w*, WHERE *b*, *n*, AND *w* DENOTE THE NUMBER OF FUNCTIONS ON WHICH AEPD-JADE PERFORMS SIGNIFICANTLY BETTER THAN, STATISTICALLY EQUIVALENT TO, AND SIGNIFICANTLY WORSE THAN THEIR PEER ALGORITHMS, RESPECTIVELY

	D=30							*D*=50						
	DE/rand/1/bin	JADE	jDE	SaDE	CoDE	Pro DE/rand/1/bin	EPSDE	DE/rand/1/bin	JADE	jDE	SaDE	CoDE	Pro DE/rand/1/bin	EPSDE
AEPD-JADE	24/1/0	19/5/1	21/4/0	24/0/1	22/1/2	24/1/0	24/1/0	22/1/2	16/4/5	19/6/0	24/0/1	19/3/3	22/1/2	18/6/1

in [1] and [4]. We use the JADE algorithm with an archive in this paper since it showed promising results compared with JADE without an archive in [30]. The parameters c and p of JADE were set to 0.1 and 0.2, which were recommended in [30], respectively. For the algorithms DE/rand/1/bin, JADE, jDE, SaDE, and Pro DE/rand/1/bin, the population size was set to $NP = 100$ for $D = 30$ and $NP = 200$ for $D = 50$, which was used in [22], [27], [28], and [30]. The population size was set to $NP = 30$, $NP = 100$, and $NP = 50$ for CoDE, HdDE, and EPSDE, respectively, as same as in their original papers. AEPD-JADE uses the same parameter settings as JADE accordingly, except that the population size for AEPD-JADE was set to $NP = 20$ for all experiments. For the CLPSO algorithm, the population size was set to $NP = \lfloor 10+2\cdot\sqrt{D}\rfloor$, which was suggested in the standard PSO 2007 (SPSO-07) [53]. For the IPOP-CMA-ES algorithm, all the parameters including the population size used the default settings in [51].

Note that, we did not test HdDE in the experiments with varying population size because HdDE was designed with two fix-sized sub-populations.

2) Performance Comparison: Table I summarizes the statistical results of average errors of involved algorithms on all problems with $D = 30$ and $D = 50$, where b, n, and w denote the number of functions on which AEPD-JADE performs significantly better than, statistically equivalent to, and significantly worse than other algorithms, respectively. In Table I, AEPD-JADE performs much better than all other algorithms in terms of the number of functions on which an algorithm performs significantly better than its peer algorithms. Note that, the results of average errors for all problems with $D = 30$ and $D = 50$ are provided in Tables II and III in the supplementary material listed in the Appendix.

We also compare all DE algorithms (except HdDE due to its population size cannot be changed) with the same population size $NP = 20$ to ensure a fair testing condition. The results of average errors for all problems with $D = 30$ and $D = 50$ are provided in Tables IV and V in the supplementary material listed in the Appendix. Table II summarizes the statistical results. In Table II, AEPD-JADE performs much better than all other algorithms in terms of the number of functions on which an algorithm performs significantly better than its peer algorithms.

B. Comparison Between Algorithms With and without AEPD

1) Comparison of Small Population Size: In this subsection, we investigate the impact of AEPD on the DE algorithms (except HdDE due to its population size cannot be changed) in case the population size is small ($NP = 6$).[2] The parameter settings are the same as the settings in Section V-A1 except for the population sizes, which were set to 6 for all the tested algorithms in this subsection.

Table III summarizes the results of average errors on each function with $D = 30$ over 30 independent runs. For each function, the best result of all algorithms is shown in bold font. From the results, AEPD-JADE performs significantly better than DE/rand/1/bin, JADE, jDE, SaDE, Pro DE/rand/1/bin, and EPSDE on all functions except for f_8 where its performance is equivalent to JADE and slightly worse than SaDE. Compared with CoDE, AEPD-JADE performs significantly better than CoDE on most functions. Using the same small population size (i.e., $NP = 6$), the other algorithms cannot achieve a good performance even though JADE, jDE, and SaDE have adaptive F and CR to adjust their population diversity.

2) Impact of Population Size: The objectives of this subsection are to check if AEPD is helpful for other DE variants and to study the impact of population size on the performance of AEPD-DE algorithms. To achieve the above objectives, we apply AEPD to four conventional and six adaptive DE algorithms, which are DE/rand/1/bin, DE/best/1/bin, DE/current-to-best/1/bin, DE/rand-to-best/1/bin [see (1)–(4)], JADE, jDE, SaDE, CoDE, Pro DE/rand/1/bin, and EPSDE, respectively. To simplify the experimental study, only the shifted sphere function f_1 (a unimodal function) and the hybrid composition function f_{15} (a multimodal function) with $D = 30$ were

[2]We chose $NP = 6$ because for CoDE [21] the smallest acceptable population size is 6.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
AVERAGE ERROR VALUES ± STANDARD DEVIATIONS ARCHIVED BY THE TESTED ALGORITHMS ON THE 25 30-DIMENSIONAL CEC05 BENCHMARK FUNCTIONS OVER 30 INDEPENDENT RUNS WITH NP = 6. b, n, AND w DENOTE THE NUMBER OF FUNCTIONS ON WHICH AEPD-JADE PERFORMS SIGNIFICANTLY BETTER THAN, STATISTICALLY EQUIVALENT TO, AND SIGNIFICANTLY WORSE THAN ITS PEER ALGORITHMS, RESPECTIVELY

F	AEPD-JADE	DE/rand/1/bin	JADE	jDE	SaDE	CoDE	Pro DE/rand/1/bin	EPSDE
f_1	5.68e-014±0.00e+000	6.08e+004±1.17e+004†	2.07e+001±1.03e+002†	2.20e+004±1.33e+004†	5.73e+004±1.67e+004†	**0.00e+000±0.00e+000**‡	6.48e+004±1.55e+004†	8.77e-009±4.79e-008†
f_2	3.74e-009±1.22e-008	8.29e+004±2.26e+004†	5.50e+002±1.44e+003†	4.32e+004±1.60e+004†	8.34e+004±2.70e+004†	**2.46e-013±6.65e-013**‡	8.87e+004±2.48e+004†	1.49e+003±1.77e+003†
f_3	5.49e-014±1.04e-014	2.24e+009±1.59e+009†	2.27e+006±1.24e+007†	6.10e+008±7.33e+008†	2.09e+009±1.50e+009†	**1.89e-015±1.04e-014**‡	2.61e+009±1.73e+009†	6.92e-008±2.88e-007†
f_4	**3.59e-002±4.51e-002**	8.68e+004±2.31e+004†	2.25e+004±1.08e+004†	4.88e+004±1.73e+004†	8.50e+004±2.07e+004†	4.18e+003±3.50e+003†	8.98e+004±2.31e+004†	3.77e+004±1.63e+004†
f_5	2.45e+001±2.90e+001	3.91e+005±4.53e+004†	1.09e+004±1.08e+004†	2.29e+005±5.74e+004†	4.01e+005±6.00e+004†	**1.75e+000±3.49e+000**‡	3.82e+005±4.34e+004†	2.80e+004±1.48e+004†
f_6	**3.27e+000±5.22e+000**	4.50e+010±2.26e+010†	1.00e+007±4.44e+007†	1.62e+010±1.41e+010†	4.50e+010±1.99e+010†	1.40e+001±1.80e+001†	5.06e+010±2.74e+010†	9.68e+001±8.83e+001†
f_7	2.11e-002±1.97e-002	1.35e+004±1.70e+003†	1.08e+001±3.40e+001†	5.34e+003±1.94e+003†	1.11e+004±2.59e+003†	**1.50e-002±1.12e-002**	1.38e+004±1.91e+003†	3.94e+000±7.64e+000†
f_8	2.09e+001±8.80e-002	2.10e+001±7.51e-002†	2.09e+001±1.18e-001	2.09e+001±5.96e-002†	**2.07e+001±1.01e-001**‡	2.09e+001±4.46e-002†	2.10e+001±7.24e-002†	2.10e+001±5.66e-002†
f_9	**4.91e+000±4.57e+000**	3.12e+002±4.88e+001†	1.15e+002±3.90e+001†	1.38e+002±4.42e+001†	3.30e+002±5.27e+001†	9.65e+000±5.35e+000†	3.23e+002±4.09e+001†	2.52e+001±1.99e+001†
f_{10}	1.34e+002±3.63e+001	4.89e+002±1.04e+002†	2.26e+002±6.18e+001†	3.00e+002±7.59e+001†	5.60e+002±1.06e+002†	**1.34e+002±3.27e+001**	5.00e+002±6.66e+001†	2.30e+002±4.61e+001†
f_{11}	**2.15e+001±3.74e+000**	3.67e+001±3.10e+000†	2.95e+001±2.24e+000†	3.03e+001±3.43e+000†	3.70e+001±4.77e+000†	3.03e+001±2.27e+000†	3.77e+001±3.05e+000†	3.09e+001±2.86e+000†
f_{12}	**4.23e+003±4.78e+003**	8.68e+005±3.63e+005†	1.37e+004±1.12e+004†	2.59e+005±1.11e+005†	8.18e+005±3.99e+005†	4.22e+004±1.18e+004†	9.77e+005±2.87e+005†	2.21e+004±1.08e+004†
f_{13}	**3.00e+000±1.36e+000**	6.51e+006±7.35e+006†	2.60e+002±6.02e+002†	1.57e+006±1.88e+006†	6.33e+006±6.54e+006†	4.55e+000±1.04e+001	9.53e+006±8.27e+006†	8.92e+001±6.04e+001†
f_{14}	**1.26e+001±4.46e-001**	1.30e+001±5.05e-001†	1.28e+001±3.57e-001†	1.28e+001±3.45e-001†	1.34e+001±3.95e-001†	1.31e+001±3.05e-001†	1.31e+001±4.95e-001†	1.32e+001±2.75e-001†
f_{15}	**3.40e+002±2.99e+002**	1.13e+003±4.60e+001†	9.65e+002±6.21e+001†	1.01e+003±4.91e+001†	1.18e+003±4.01e+001†	7.32e+002±3.36e+001†	1.14e+003±3.25e+001†	8.16e+002±6.45e+001†
f_{16}	7.92e+002±3.87e+001	1.13e+003±5.33e+001†	9.49e+002±7.59e+001†	1.00e+003±5.09e+001†	1.17e+003±3.67e+001†	**7.54e+002±3.61e+001**‡	1.13e+003±5.02e+001†	8.91e+002±4.37e+001†
f_{17}	7.91e+002±3.64e+001	1.14e+003±5.34e+001†	9.88e+002±6.12e+001†	1.03e+003±5.02e+001†	1.14e+003±7.45e+001†	**7.88e+002±3.94e+001**	1.16e+003±5.09e+001†	1.01e+003±6.98e+001†
f_{18}	1.20e+003±3.52e+001	1.35e+003±2.48e+001†	1.30e+003±2.00e+001†	1.29e+003±3.63e+001†	1.37e+003±3.48e+001†	**1.15e+003±9.85e+001**‡	1.35e+003±2.58e+001†	1.31e+003±2.66e+001†
f_{19}	1.16e+003±5.92e+001	1.33e+003±2.40e+001†	1.27e+003±2.84e+001†	1.26e+003±3.94e+001†	1.35e+003±3.09e+001†	**1.10e+003±1.68e+002**‡	1.34e+003±3.11e+001†	1.27e+003±4.49e+001†
f_{20}	**1.12e+003±3.20e+001**	1.27e+003±2.91e+001†	1.20e+003±3.07e+001†	1.22e+003±2.96e+001†	1.31e+003±2.67e+001†	1.14e+003±5.04e+001†	1.29e+003±2.59e+001†	1.23e+003±3.92e+001†
f_{21}	**1.26e+003±3.10e+001**	1.43e+003±4.72e+001†	1.33e+003±3.40e+001†	1.35e+003±4.35e+001†	1.45e+003±3.35e+001†	1.28e+003±3.54e+001†	1.43e+003±3.44e+001†	1.31e+003±3.67e+001†
f_{22}	**1.17e+003±4.48e+001**	1.27e+003±3.26e+001†	1.19e+003±3.47e+001†	1.20e+003±3.49e+001†	1.25e+003±3.81e+001†	1.21e+003±2.36e+001†	1.26e+003±3.96e+001†	1.24e+003±2.35e+001†
f_{23}	1.29e+003±2.19e+001	1.40e+003±3.65e+001†	1.32e+003±2.70e+001†	1.36e+003±3.34e+001†	1.45e+003±3.68e+001†	**1.28e+003±3.79e+001**	1.41e+003±3.66e+001†	1.31e+003±3.64e+001†
f_{24}	**1.09e+003±1.41e+001**	1.18e+003±2.25e+001†	1.13e+003±1.49e+001†	1.14e+003±2.16e+001†	1.18e+003±2.47e+001†	1.11e+003±1.18e+001†	1.18e+003±2.34e+001†	1.13e+003±1.77e+001†
f_{25}	**1.18e+003±2.39e+001**	1.29e+003±2.10e+001†	1.22e+003±1.86e+001†	1.24e+003±2.48e+001†	1.28e+003±2.97e+001†	1.24e+003±2.50e+001†	1.28e+003±2.35e+001†	1.24e+003±1.95e+001†
$b/n/w$	—	25/0/0	24/1/0	25/0/0	24/0/1	13/5/7	25/0/0	25/0/0

The symbols † and ‡ denote that AEPD-JADE performs significantly better than and worse than this algorithm at a 0.05 level of significance by the paired samples Wilcoxon signed rank test.

used. The parameter settings are the same as the settings in Section V-A1 except that the population size for each algorithm varies from NP = 6 to NP = 100. Fig. 3 shows the comparison of average errors over 30 independent runs for each pair of algorithms with different population sizes.

From Fig. 3, it can be seen that AEPD does make the tested algorithms much less sensitive to the parameter of population size. The extended algorithm in each pair of algorithms performs much better than the original algorithm except the cases of CoDE where there is no significantly difference on f_1 and EPSDE where the performance gets worse on f_1 when NP = 6. For EPSDE, a small population is sufficient to solve the unimodal function f_1. In this case, the rediversification brought by AEPD would slow down the convergence of population, so AEPD-EPSDE performs worse than EPSDE on f_1 when NP = 6. The performance gets worse as population size increases for AEPD-SaDE on f_{15}. A similar trend is also seen for AEPD-CoDE and AEPD-EPSDE on these two functions. Therefore, population size should be small for algorithms using AEPD. This is because in AEPD population diversity is enhanced by rediversification rather than by the introduction of new individuals.

Different from other algorithms where population diversity is enhanced by introducing new individuals, the AEPD mechanism is able to diversify the population automatically either to further exploit the same area or to explore new promising areas as necessary. Therefore, a small population size may not affect the exploitation/exploration of AEPD. This helps avoid using a large population, which in turns may increase computational cost. This advantage of AEPD will be further investigated in the following subsection.

3) Comparison of Ineffective Moves: For an individual, an ineffective move is a move to its trial vector that does not improve its fitness. The smaller number of ineffective moves an algorithm generates, the more better individuals the algorithm generates. In order to show the improvement of AEPD for an algorithm regarding reducing the ineffective moves, we compute a measure named the ineffective moving ratio. This measure is calculated as the ratio of the number of ineffective moves to the number of all moves generated by DE operators over all generations.

An experimental study was performed on seven pairs of algorithms (DE/rand/1/bin ± AEPD, JADE ± AEPD, jDE ± AEPD, SaDE ± AEPD, CoDE ± AEPD, Pro DE/rand/1/bin ± AEPD, and EPSDE ± AEPD) on each function with D = 30. The parameter settings of all algorithms are the same as used in Section V-A1. Table IV summarizes the mean ratios over 30 independent runs. s_1 is the mean ratio of the original algorithms with NP = 20; s_2 is the mean ratio of the original algorithms with their original NP values (see Section V-A1); and s_3 is the mean ratio of the original algorithms plus AEPD and NP = 20. For each algorithm, the smallest value on each function is shown in bold font. Table V summarizes the statistical results of Table IV, where b, n and w denote the number of functions on which the value of s in the first column is significantly smaller than, statistically equivalent to, and significantly larger than the value of s in the current row, respectively.

The experimental results show that for JADE, jDE, CoDE, and EPSDE algorithms, the population prematurely converged on most functions, when NP = 20. Therefore, for these four algorithms, in Table IV, the average values of s_1 are larger than s_2. Table V also confirms that this difference is significance in most most cases. However, when these four algorithms are combined with AEPD, their ratios of ineffective moves (s_3) decrease significantly. The results show that

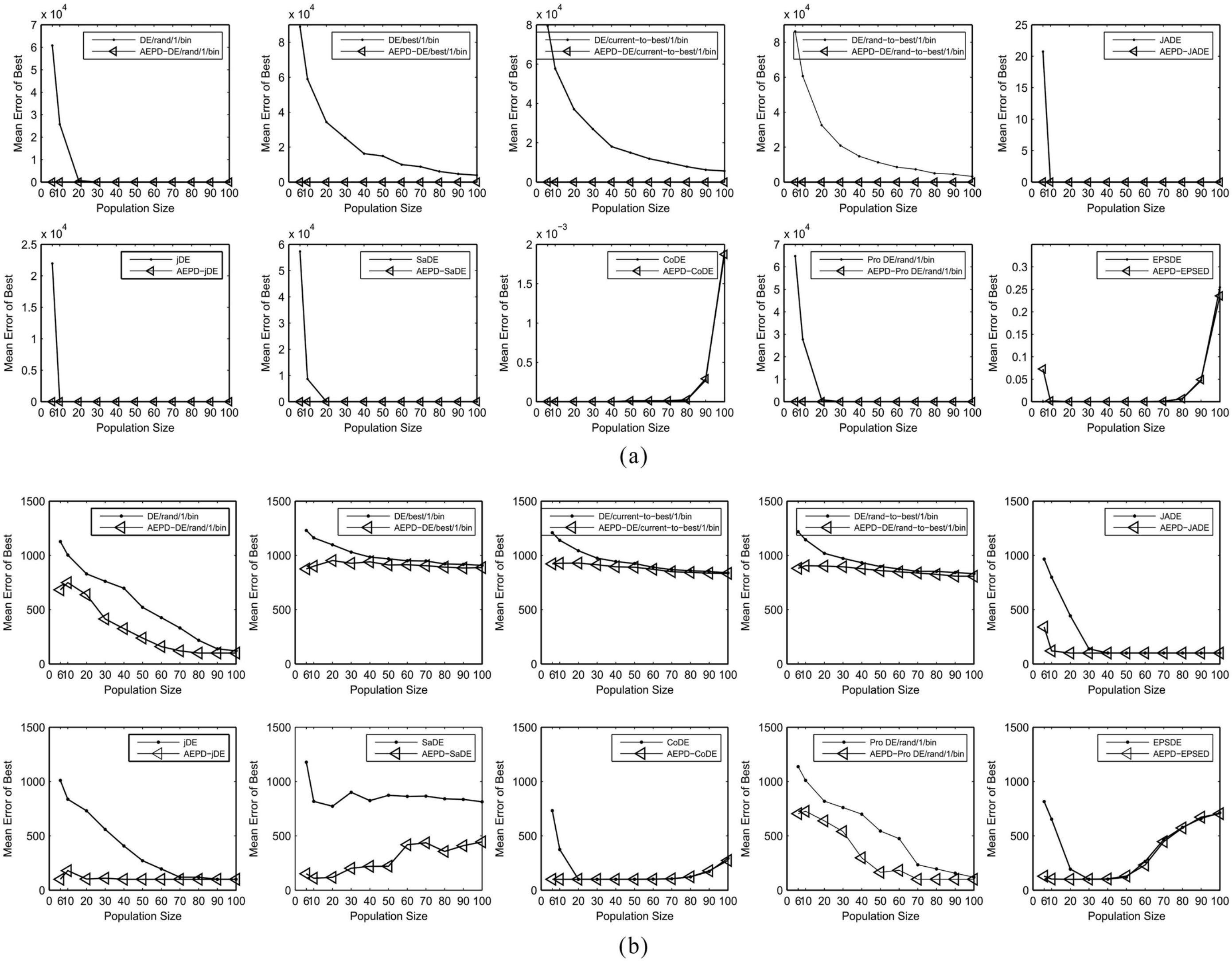

Fig. 3. Average error results of the ten pairs of algorithms with different population sizes on 30-dimensional f_1 and f_{15} over 30 independent runs. (a) Shifted sphere function f_1. (b) Hybrid composition function f_{15}.

AEPD enables population to continue to evolve by diversifying the prematurely converged population. It should be noted that increasing population size may increase the number of ineffective moves, as can be seen from the comparisons in Table IV and V between s_1 and s_2 for DE/rand/1/bin, SaDE, and Pro DE/rand/1/bin algorithms. Therefore, we suggest that the population size of algorithms with AEPD should be small.

Intuitively, the more better individuals an algorithm generates, the more possibly its population moves close to the global optimum. However, our experimental results show that the best solution got by an algorithm with a small ineffective moving ratio is not absolutely better than the best solution got by an algorithm with a large ineffective moving ratio. For an algorithm, a small ineffective moving ratio indicates that it has a good search capability.

C. Sensitivity Analysis of the Parameter T of AEPD

The parameter T determines the value of ω_j , which is the threshold to check if population diversity is poor in the jth dimension [see (8a)]. The choice of T would affect the performance of algorithms with AEPD, and it may also relate to the choice of the population size. To find out a good choice of T, we compare the performance of AEPD-JADE algorithm with different combinations of $T = \{1, 10^{-1}, 10^{-2}, \ldots, 10^{-6}\}$ and $NP = \{6, 10, 20, \ldots, 100\}$ on the hybrid composition function f_{15} (a multimodal function). Note that, the value of T is expected below 1 as the diversity threshold ω_j should be small enough to make sure it can check that the population has been converging in a dimension. The other parameter settings of AEPD-JADE are the same as the settings in Section V-A1. Table VI summarizes the results of average errors over 30 independent runs. For each NP setting, the best result of algorithms with different values of T is shown in bold font.

From the results, two observations can be seen. Firstly, population size does affect the choice of T on the performance of AEPD-JADE. However, the affection is obvious only when population size is small (e.g., $NP < 20$). Under a same value of T, the mean errors are almost the same for different

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV

AVERAGE RATIOS OF INEFFECTIVE MOVES ON THE 25 30-DIMENSIONAL CEC05 BENCHMARK FUNCTIONS OVER 30 INDEPENDENT RUNS WHERE s_1, s_2, AND s_3 ARE THE RATIO SCORES OF THE ORIGINAL ALGORITHMS WITH NP = 20, THE ORIGINAL ALGORITHMS WITH ORIGINAL NP VALUES (SECTION V-A1), AND THE ORIGINAL ALGORITHM PLUS AEPD AND NP = 20, RESPECTIVELY

F	DE/rand/1/bin			JADE			jDE			SaDE			CoDE			Pro DE/rand/1/bin			EPSDE		
	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
f_1	**0.750**	0.923	0.827	0.977	**0.908**	0.952	0.980	0.905	**0.762**	**0.425**	0.914	0.444	0.970	0.955	**0.938**	**0.768**	0.923	0.835	0.963	0.938	**0.937**
f_2	**0.703**	0.955	0.706	0.875	0.844	**0.831**	0.928	0.972	**0.918**	**0.326**	0.353	0.337	**0.962**	0.980	0.962	**0.699**	0.955	0.703	0.976	0.982	**0.975**
f_3	**0.760**	0.908	0.777	0.972	**0.884**	0.940	0.976	0.886	**0.751**	0.409	0.894	**0.406**	0.964	0.945	**0.926**	0.777	0.908	**0.748**	0.955	0.937	**0.936**
f_4	0.988	0.969	**0.968**	0.980	**0.896**	0.960	0.986	0.982	**0.972**	0.995	0.965	**0.961**	0.989	0.993	**0.987**	0.986	0.968	**0.968**	0.983	**0.945**	0.949
f_5	**0.635**	0.905	0.635	0.650	**0.618**	0.654	**0.748**	0.906	0.760	**0.349**	0.425	0.366	**0.968**	0.979	0.969	**0.632**	0.905	0.633	**0.945**	0.960	0.947
f_6	0.644	0.852	**0.622**	0.760	0.744	**0.696**	0.762	0.872	**0.753**	**0.331**	0.370	0.348	0.946	0.953	**0.946**	0.643	0.852	**0.621**	0.947	0.948	**0.943**
f_7	0.677	0.913	**0.620**	0.929	**0.864**	0.906	0.939	0.894	**0.816**	0.350	0.822	**0.343**	0.971	0.973	**0.959**	0.681	0.913	**0.619**	0.951	0.953	**0.946**
f_8	0.999	0.997	**0.974**	**0.968**	0.994	0.974	0.999	0.997	**0.974**	0.999	0.997	**0.974**	0.998	0.999	**0.988**	0.999	0.997	**0.974**	0.999	0.999	**0.984**
f_9	**0.722**	0.993	0.811	0.973	**0.866**	0.939	0.980	0.940	**0.804**	0.769	0.837	**0.603**	0.971	0.957	**0.954**	**0.721**	0.993	0.797	0.965	0.974	**0.962**
f_{10}	0.730	0.992	**0.714**	0.985	0.968	**0.929**	0.978	0.990	**0.873**	0.401	0.764	**0.393**	0.998	0.997	**0.989**	**0.714**	0.993	0.740	0.995	0.992	**0.976**
f_{11}	0.918	0.997	**0.878**	0.996	0.989	**0.937**	0.961	0.996	**0.931**	**0.602**	0.604	0.717	0.999	0.998	**0.989**	0.950	0.997	**0.878**	0.998	0.997	**0.979**
f_{12}	**0.624**	0.960	0.629	0.974	0.961	**0.793**	**0.768**	0.985	0.770	**0.390**	0.504	0.433	0.996	0.995	**0.987**	**0.622**	0.963	0.627	0.991	0.985	**0.966**
f_{13}	**0.626**	0.945	0.652	0.969	0.950	**0.855**	0.844	0.969	**0.840**	**0.341**	0.395	0.414	0.992	0.989	**0.982**	**0.622**	0.949	0.651	0.986	0.974	**0.963**
f_{14}	0.988	0.997	**0.974**	**0.975**	0.994	0.977	0.999	0.997	**0.975**	0.884	0.847	**0.751**	0.999	0.999	**0.989**	0.985	0.997	**0.975**	0.999	0.998	**0.983**
f_{15}	0.902	0.885	**0.739**	0.952	**0.770**	0.827	0.969	0.833	**0.772**	0.925	0.872	**0.662**	0.948	0.941	**0.934**	0.908	0.883	**0.739**	0.949	0.960	**0.948**
f_{16}	0.829	0.869	**0.740**	0.980	**0.833**	0.844	0.963	0.837	**0.834**	0.640	0.830	**0.595**	0.991	0.994	**0.968**	0.827	0.869	**0.763**	0.992	0.986	**0.965**
f_{17}	0.996	0.973	**0.966**	0.995	0.976	**0.953**	0.995	0.983	**0.963**	0.997	0.981	**0.971**	0.998	0.997	**0.991**	0.996	0.974	**0.968**	0.995	0.988	**0.956**
f_{18}	**0.727**	0.870	0.811	0.993	**0.852**	0.871	0.927	0.839	**0.835**	0.667	**0.561**	0.611	0.996	0.996	**0.971**	**0.733**	0.870	0.824	0.998	0.996	**0.980**
f_{19}	**0.729**	0.871	0.810	0.991	**0.854**	0.872	0.928	0.839	**0.828**	0.618	**0.552**	0.622	0.996	0.997	**0.971**	**0.712**	0.871	0.807	0.998	0.996	**0.980**
f_{20}	**0.717**	0.874	0.827	0.995	0.975	**0.935**	0.906	**0.845**	0.921	0.691	**0.544**	0.622	0.998	0.997	**0.979**	**0.711**	0.875	0.833	0.997	0.995	**0.980**
f_{21}	0.823	0.959	**0.793**	0.986	0.932	**0.923**	0.972	0.971	**0.899**	0.613	0.851	**0.569**	0.998	0.997	**0.990**	0.829	0.958	**0.802**	0.996	0.993	**0.978**
f_{22}	0.998	0.987	**0.887**	0.995	0.993	**0.948**	0.998	0.994	**0.917**	0.996	0.964	**0.963**	0.999	0.998	**0.990**	0.998	0.981	**0.896**	0.999	0.997	**0.983**
f_{23}	0.993	0.971	**0.955**	0.996	**0.984**	0.987	0.992	0.977	**0.937**	0.994	0.967	**0.874**	0.998	0.997	**0.993**	0.992	0.970	**0.954**	0.997	0.994	**0.989**
f_{24}	0.996	0.993	**0.963**	0.996	0.988	**0.973**	0.997	0.992	**0.971**	0.995	0.967	**0.933**	0.998	0.998	**0.989**	0.995	0.993	**0.962**	0.991	0.984	**0.925**
f_{25}	0.991	0.986	**0.931**	0.995	0.988	**0.960**	0.990	0.995	**0.920**	0.994	0.961	**0.916**	0.999	0.999	**0.989**	0.990	0.984	**0.922**	0.991	0.985	**0.921**
average	0.819	0.942	0.808	0.954	0.905	0.897	0.939	0.936	0.868	0.668	0.750	0.633	0.986	0.985	0.973	0.820	0.942	0.810	0.982	0.978	0.962

TABLE V

STATISTICAL RESULTS OF TABLE IV: $b/n/w$, WHERE b, n, AND w DENOTE THE NUMBER OF FUNCTIONS ON WHICH THE VALUE OF s IN THE FIRST COLUMN IS SIGNIFICANTLY SMALLER THAN, STATISTICALLY EQUIVALENT TO, AND SIGNIFICANTLY LARGER THAN THE VALUE OF s IN THE CURRENT ROW, RESPECTIVELY

	DE/rand/1/bin			JADE			jDE			SaDE			CoDE			Pro DE/rand/1/bin			EPSDE		
	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
s_1	—	16/2/7	8/4/13	—	2/1/22	2/1/22	—	8/1/16	2/3/20	—	11/3/11	7/7/11	—	8/1/16	1/2/22	—	16/2/7	8/5/12	—	4/2/19	1/2/22
s_2	7/2/16	—	0/1/24	22/1/2	—	9/3/13	16/1/8	—	1/3/21	11/3/11	—	4/2/19	16/1/8	—	0/0/25	7/2/16	—	0/1/24	19/2/4	—	1/0/24
s_3	13/4/8	24/1/0	—	22/1/2	13/3/9	—	20/3/2	21/3/1	—	11/7/7	19/2/4	—	22/2/1	25/0/0	—	12/5/8	24/1/0	—	22/2/1	24/0/1	—

TABLE VI

AVERAGE ERROR VALUES ± STANDARD DEVIATIONS ARCHIVED BY THE AEPD-JADE ALGORITHMS WITH DIFFERENT VALUES OF T AND NP ON 30-DIMENSIONAL f_{15} OVER 30 INDEPENDENT RUNS

	T=1	T=10^{-1}	T=10^{-2}	T=10^{-3}	T=10^{-4}	T=10^{-5}	T=10^{-6}
NP=6	**1.00e+002±3.14e-013**	1.40e+002±1.51e+002	1.80e+002±2.06e+002	3.40e+002±2.99e+002	4.19e+002±3.04e+002	4.17e+002±3.02e+002	4.35e+002±2.98e+002
NP=10	**1.00e+002±3.72e-013**	1.19e+002±1.05e+002	1.20e+002±1.08e+002	1.20e+002±1.10e+002	1.60e+002±1.82e+002	1.81e+002±2.09e+002	2.17e+002±2.38e+002
NP=20	1.00e+002±2.89e-014	1.00e+002±2.89e-014	1.00e+002±2.89e-014	1.00e+002±2.89e-014	**1.00e+002±2.06e-013**	1.00e+002±2.89e-014	1.16e+002±8.61e+001
NP=30	1.00e+002±4.83e-013	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**
NP=40	1.00e+002±3.11e-012	1.00e+002±3.47e-013	1.00e+002±2.80e-013	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**	**1.00e+002±2.89e-014**
NP=50	1.00e+002±5.07e-010	1.00e+002±9.50e-013	1.00e+002±3.72e-013	1.00e+002±1.69e-013	1.00e+002±2.80e-013	**1.00e+002±2.89e-014**	1.00e+002±2.33e-013
NP=60	1.00e+002±1.80e-008	1.00e+002±2.31e-011	1.00e+002±6.60e-012	1.00e+002±4.25e-013	**1.00e+002±3.17e-013**	**1.00e+002±3.17e-013**	1.00e+002±4.10e-013
NP=70	1.00e+002±4.44e-008	1.00e+002±1.30e-009	1.00e+002±2.87e-011	1.00e+002±2.11e-011	**1.00e+002±5.37e-013**	1.00e+002±4.62e-012	1.00e+002±6.03e-013
NP=80	1.00e+002±5.44e-007	1.00e+002±4.13e-009	1.00e+002±1.73e-009	1.00e+002±7.44e-011	1.00e+002±2.06e-010	1.00e+002±9.58e-011	**1.00e+002±6.75e-012**
NP=90	1.00e+002±1.01e-006	1.00e+002±2.70e-008	1.00e+002±9.58e-009	1.00e+002±4.04e-009	1.00e+002±5.04e-009	1.00e+002±7.73e-010	**1.00e+002±8.32e-012**
NP=100	1.00e+002±2.80e-005	1.00e+002±5.97e-007	1.00e+002±2.08e-008	1.00e+002±5.68e-008	1.00e+002±1.18e-008	1.00e+002±9.64e-010	**1.00e+002±1.14e-011**

population sizes when $NP \geq 20$. Secondly, under a same value of NP when $NP \geq 20$, there is no significant difference in the mean errors with different values of T. In addition, the errors in the cases that $NP \geq 20$ are much better than the errors in the cases that $NP < 20$. According to the results in Table VI, we suggest $T = 10^{-3}$ for algorithms with AEPD.

D. Working Mechanism of AEPD

The above experimental studies show that AEPD does improve the performance of classic DE variants. In this subsection, we will investigate the working mechanism of AEPD through the AEPD-DE/rand/1/bin algorithm on the Ackley function, a multimodal and nonseparable function, which is as follows:

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + \exp(1),\ x_i \in [-32, 32]. \tag{22}$$

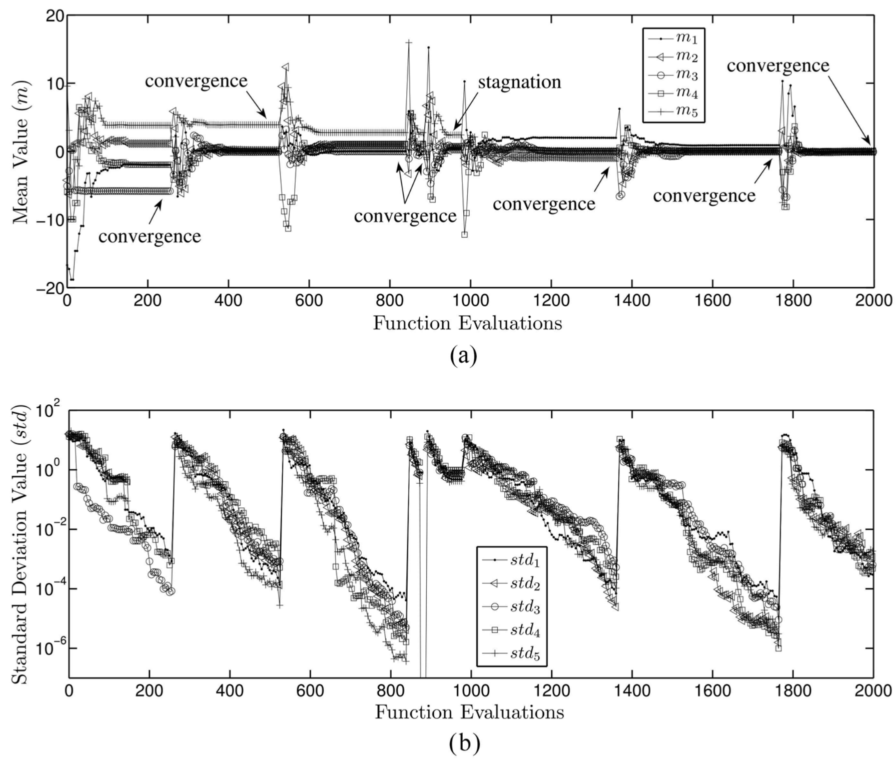

Fig. 4. Working mechanisms of AEPD by the AEPD-DE/rand/1/bin algorithm on the 5-dimensional Ackley function, when NP = 5. (a) Mean value of individual variables in each dimension. (b) Standard deviation value of individual variables in each dimension.

The global optimum of this function is $\mathbf{x}=(0,\ldots,0)$.

In order to further investigate the working mechanism of AEPD, we present the result of AEPD-DE/rand/1/bin on the Ackley function with D = 5. Fig. 4(a) and (b) show the progress of the algorithm for m_j (6) and std_j (7), respectively. The vertical axis of Fig. 4(b) is the logarithmic value of std_j, so the curve will be broken when std_j decreases to 0.

After the evolutionary process begins, the population gradually converges: it is able to converge nearby the global optimum in three of the five dimensions after 255 function evaluations [see Fig. 4(a)]. The evolution speed of the population is not the same in all dimensions. When the number of evaluations reaches 255, the value of std_j in each dimension decreases below the value of ω_j [see Fig. 4(b)]. Therefore, the population is regenerated. Accordingly, for each dimension, the mean value m_j changes and the standard deviation std_j increases. After the first population regeneration the population converges around the global optimum in four of the five dimensions when the number of evaluations reaches 524. The values of std_j decrease below the value of ω_j again. Thereafter, another population regeneration occurs. The population repeats the above process. After a few generations, the population stagnates when the number of evaluations reaches 976, where the values of m_j and std_j in all dimensions remain unchanged. The stagnation status is identified by AEPD and the population is regenerated again. After several generations, the population converges around the global optimum. From Fig. 4(a), it can be seen that for all dimensions the population is able to converge gradually to the global optimum on the tested function.

The experimental results show that, based on the current population status and the evolving history, AEPD-DE/rand/1/bin is able to adaptively adjust its population diversity to appropriate levels at appropriate moments on the tested function.

VI. Conclusion

In order to improve the performance of DE algorithms, this paper proposes a novel population adaptation approach (AEPD) that enables DE algorithms to adaptively enhance their population diversity to stop the population from being converged or stagnated. Different from existing traditional population adaptation methods that aim to adjust population diversity by tuning population size, the proposed approach does not change population size but diversifies a population. And the diversification is implemented at the dimensional level, i.e., the population is diversified only in the dimensions that need to be diversified.

A set of 25 CEC2005 benchmark functions are selected to compare the performance between the DE algorithms with AEPD and other peer algorithms. From the experimental results in this paper, several conclusions can be drawn.

Firstly, from the experimental investigation on the working mechanism of AEPD, it can be seen that AEPD is able to relieve the premature and stagnation problem to some extent. By analyzing the current population and previous learning, AEPD can auto-diversify the population when it is identified as converged or stagnated. Secondly, AEPD is able to enhance the performance of DE algorithms at least on the

JADE algorithms, in particular when the search space is large. Thirdly, AEPD is also able to make algorithms much less sensitive to the parameter of population size, so the algorithms with AEPD can also be competitive even if the population size is small. Fourthly, the fact that the evolutionary progress of a population in different dimensions is not strictly synchronous should be addressed in the future. Finally, the ineffective moving ratio can be reduced by the AEPD approach on most tested algorithms. And interestingly, reducing the ineffective moving ratio may be a promising research method to improve the performance of DE algorithms. In addition, the AEPD approach can be used in any existing DE algorithms and even other kinds of real-coded EAs, such as PSO and GA.

For future work of this paper, two interesting issues should be addressed for AEPD-DE algorithms. The first one is that explored areas in the search space should be avoided to save computing resources. Applying the AEPD-DE algorithms to real-world problems is also an interesting topic.

Appendix
Supplementary Material Available on the Web

The detailed experimental results in the supplementary material consist of the following two parts.

1) The comparison of AEPD-JADE with other algorithms: the results of average errors over 30 independent runs on each function with dimensions $D = 30$ and $D = 50$.
2) The sensitivity studies of the parameter c in (14a) and the parameter a in (19): the results of average errors over 30 independent runs on each function with dimension $D = 30$.

These results are available at http://cs.cug.edu.cn/teacherweb/yangming/documents/AEPD-supplement.pdf.

Acknowledgment

The authors would like to thank Dr. T. T. Nguyen at the School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, U.K., for carefully proofreading this paper and they would also like to thank Prof. X. Yao at the School of Computer Science, University of Birmingham, U.K., for suggestions to improve this paper.

References

[1] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Opt.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
[2] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.
[3] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, Nimes, France, Apr. 2004, pp. 165–170.
[4] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. 8th Annu. Conf. Genet. Evol. Comput. (GECCO)*, New York, NY, USA, 2006, pp. 485–492.
[5] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proc. WSEAS Int. Conf. Adv. Intell. Syst. Fuzzy Syst. Evol. Comput.*, Vancouver, BC, Canada, 2002, pp. 293–298.
[6] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proc. 8th Int. Conf. Soft Comput.*, Brno, Czech Republic, 2002, pp. 11–18.
[7] R. Mallipeddi and P. Suganthan, "Empirical study on the effect of population size on differential evolution algorithm," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, Hong Kong, Jun. 2008, pp. 3663–3670.
[8] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 61–106, Feb. 2010.
[9] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[10] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1149–1155, 2009.
[11] N. Padhye, P. Mittal, and K. Deb, "Differential evolution: Performances and analyses," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 1960–1967.
[12] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. 6th Int. Mendel Conf. Soft Comput.*, Shenyang, China, 2000, pp. 76–83.
[13] J. Teo, "Differential evolution with self-adaptive populations," in *Proc. 9th Int. Conf. Knowl.-Based Intell. Inf. Eng. Syst. (KES)*, Melbourne, VIC, Australia, 2005, pp. 1284–1290.
[14] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Appl. Intell.*, vol. 29, no. 3, pp. 228–247, 2008.
[15] F. Neri and V. Tirronen, "On memetic differential evolution frameworks: A study of advantages and limitations in hybridization," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, Hong Kong, 2008, pp. 2135–2142.
[16] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Opt.*, vol. 27, no. 1, pp. 105–129, 2003.
[17] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.
[18] 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.
[19] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact differential evolution," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 32–54, Feb. 2011.
[20] R. Mallipeddi and P. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Proc. Swarm Evol. Memet. Comput.*, Chennai, India, 2010, pp. 71–78.
[21] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.
[22] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.
[23] B. Dorronsoro and P. Bouvry, "Improving classical and decentralized differential evolution with new mutation operator and population topologies," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 67–98, Feb. 2011.
[24] N. Padhye, P. Bhardawaj, and K. Deb, "Improving differential evolution through a unified approach," *J. Global Opt.*, vol. 55, no. 4, pp. 771–799, 2013.
[25] K. Deb and N. Padhye, "Enhancing performance of particle swarm optimization through an algorithmic link with genetic algorithms," *Comput. Optim. Appl.*, vol. 57, no. 3, pp. 761–794, 2014.
[26] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.
[27] A. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Cong. Evol. Comput. (CEC)*,Edinburgh, UK, 2005, pp. 1785–1791.
[28] 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.
[29] J. Brest, B. Boskovic, V. Z. S. Greiner, and M. Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.*, vol. 11, no. 7, pp. 617–629, 2007.
[30] J. Zhang and C. Arthur, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[31] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Computing*, vol. 15, no. 11, pp. 2141–2155, 2011.
[32] R. Ursem, "Diversity-guided evolutionary algorithms," in *Proc. Parallel Prob. Solving Nature (PPSN VII)*, Granada, Spain, 2002, pp. 462–471.

[33] F. Herrera and M. Lozano, "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," in *Genetic Algorithm Soft Computing*, Heidelberg, Germany: Physica-Verlag, 1996, pp. 95–125.

[34] A. Barker and W. Martin, "Dynamics of a distance-based population diversity measure," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, vol. 2. La Jolla, CA, USA, 2000, pp. 1002–1009.

[35] O. Olorunda and A. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, Hong Kong, 2008, pp. 1128–1134.

[36] M. Yang, Z. Cai, C. Li, and J. Guan, "An improved adaptive differential evolution algorithm with population adaptation," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. (GECCO)*, New York, NY, USA, 2013, pp. 145–152.

[37] M. Gouvêa and A. F. R. Araújo, "Diversity control based on population heterozygosity dynamics," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, Hong Kong, 2008, pp. 3671–3678.

[38] M. Wineberg and F. Oppacher, "The underlying similarity of diversity measures used in evolutionary computation," in *Proc. Int. Conf. Genet. Evol. Comput.*, Chicago, IL, USA, 2003, pp. 1493–1504.

[39] Y. Shi and R. Eberhart, "Population diversity of particle swarms," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, Hong Kong, 2008, pp. 1063–1067.

[40] R. W. Morrison and K. A. De Jong, "Measurement of population diversity," in *Proc. Euro. Conf. Artif. Evol.*, Le Creusot, France, 2002, pp. 31–41.

[41] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Review and study of genotypic diversity measures for real-coded representations," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 695–710, Oct. 2012.

[42] D. Zaharie, "Critical values for the control parameters of differential evolution algorithm," in *Proc. 8th Int. Mendel Conf. Soft Comput.*, Brno, Czech Republic, Jun. 2002, pp. 62–67.

[43] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proc. Mendel 2003 9th Int. Conf. Soft Comput.*, Brno, Czech Republic, pp. 41–46.

[44] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Braga, Portugal, 2005, pp. 2808–2815.

[45] J. Maresky, Y. Davidor, D. Gitler, G. Aharoni, and A. Barak, "Selectively destructive re-start," in *Proc. 6th Int. Conf. Genet. Algorith.*, San Francisco, CA, USA, 1995, pp. 144–150.

[46] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[47] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.

[48] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Proc. Appl. Evol. Comput.*, Jinan, China, 2004, pp. 489–500.

[49] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," School Electr. Electron. Eng., Nanyang Technol. Univ., Singapore, Tech. Rep. 2005005.

[50] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[51] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Cong. Evol. Comput. (CEC)*, vol. 2, Edinburgh, U.K., Sep. 2005, pp. 1769–1776.

[52] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.

[53] C. Maurice. (2010, Sep.). *Standard PSO 2007 (SPSO-07)* [Online]. Available: http://www.particleswarm.info/standard_pso_2007.zip

Ming Yang received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the China University of Geosciences, Wuhan, China, in 2005, 2008, and 2012, respectively.

He is currently a Lecturer with the School of Computer Science, China University of Geosciences. His current research interests include evolutionary computation and dynamic optimization.

Changhe Li (M'14) received the B.Sc. and M.Sc. degrees in computer science from the China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from the University of Leicester, Leicester, U.K., in 2011.

He is currently an Associate Professor with the School of Computer Science, China University of Geosciences. His current research interests include evolutionary algorithms, particle swarm optimization, and dynamic optimization.

Zhihua Cai received the B.Sc. degree from Wuhan University, Wuhan, China, in 1986, the M.Sc. degree from the Beijing University of Technology, Beijing, China, in 1992, and the Ph.D. degree from the China University of Geosciences, Wuhan, in 2003.

He is currently a Professor with the School of Computer Science, China University of Geosciences. His current research interests include data mining, machine learning, evolutionary computation, and their applications.

Jing Guan received the B.Sc. and M.Sc. degrees in computer science from the China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree from the Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, in 2012.

She is currently with the China Ship Development and Design Center, Wuhan. Her current research interests include evolutionary computation and image processing.