Appl Intell (2015) 42:369–388
DOI 10.1007/s10489-014-0592-3

Adaptive differential evolution with directional strategy and cloud model

Jin Gou · Wang-Ping Guo · Feng Hou · Cheng Wang · Yi-Qiao Cai

Published online: 16 October 2014

Abstract Recently, many studies have focused on differential evolution (DE), which is arguably one of the most powerful stochastic real-parameter optimization algorithms. Prominent variants of this approach largely optimize the DE algorithm; however, almost all the DE algorithms still suffer from problems like difficult parameter setting, slow convergence rate, and premature convergence. This paper presents a novel adaptive DE algorithm by constructing a trial vector generation pool, and dynamically setting control parameters according to current fitness information. The proposed algorithm adopts a distributed topology, which means the whole population is divided into three subgroups with different mutation and crossover operations used for each subgroup. However, a uniform selection strategy is employed. To improve convergence speed, a directional strategy is introduced based on the greedy strategy, which means that an individual with good performance can evolve rapidly in the optimal evolution direction. It is well known that the faster an algorithm converges, the greater the probability of premature convergence. Aimed at solving the local optimum problem, the proposed algorithm introduces a new mathematical tool in the selection process, called the membership cloud model. In essence, the cloud model improves the diversity of the population by randomly generating cloud droplets. Experimental results from executing typical benchmark functions show high quality performance of the proposed algorithm in terms of convergence, stability, and precision. They also indicate that this improved differential evolutionary algorithm can overcome the shortcoming of conventional differential evolutionary algorithms of low efficiency, while effectively avoiding falling into a local optimum.

Keywords Differential evolution · Dynamic adjustment strategy · Directional strategy · Membership cloud model

J. Gou (✉) · W.-P. Guo · F. Hou · C. Wang · Y.-Q. Cai
College of Computer Science and Technology,
Huaqiao University, Xiamen 361021, China
e-mail: goujin@gmail.com

W.-P. Guo
e-mail: winboy1988@gmail.com

F. Hou
e-mail: ted329868324@hqu.edu.cn

C. Wang
e-mail: wangcheng@hqu.edu.cn

Y.-Q. Cai
e-mail: yiqiao00@163.com

1 Introduction

Since its introduction in 1995 by Storn and Price [22], the differential evolution (DE) algorithm has been used as a simple yet powerful search technique for solving complex nonlinear and non-differentiable continuous functions. It belongs to the class of stochastic optimization algorithms, which are used to find the best-suited solution to a problem by minimizing an objective function, which is a mapping from a parameter vector $\vec{X} \in \Re^D$ to $\Re$, within the given constraints and flexibilities.

Starting with a population initialized randomly, the DE algorithm uses simple mutation and crossover operators to generate new candidate solutions, and adopts a one-to-one competitive scheme to determine whether the offspring should replace their parents in the next generation [19]. Owing to its ease of implementation and simplicity, DE has attracted much attention from researchers all over the world, resulting in many variations of the basic algorithm

with improved performance. The strategies of the variants often represent varying search capabilities in different search phases of the evolution process.

Most of the approaches to improve the standard DE algorithm mainly concentrate on four aspects: the structure and size of the population, associated control parameter setting, trial vector generation, and hybrid strategies. Of these, control parameter setting and trial vector generation directly affect the search accuracy and convergence speed of the DE algorithm, which is why associated control parameter setting is generally considered together with the trial vector generation strategy. Qin et al. [21] developed a self-adaptive DE (*SaDE*) algorithm for constrained real-parameter optimization, in which both the trial vector generation strategy and the associated control parameter values are gradually self-adapted according to the learning experience. This algorithm performs much better than both the traditional DE algorithm and several state-of-the-art adaptive parameter DE variants, such as the *ADE* [32], *SDE* [18], and *JDE* [2] algorithms. In [19], a self-adaptive DE algorithm, called *SspDE*, is proposed with each target individual having its own trial vector generation strategy, scaling factor F, and crossover rate CR, which gradually self-adapt from previous experience in generating promising solutions. Mallipeddi et al. [17] employed an ensemble of mutation strategies and control parameters in their DE algorithm (called *EPSDE*), in which a pool of distinct mutation strategies coexists with a pool of values for each control parameter throughout the evolution process competing to produce offspring. In [7], the authors designed a heterogeneous distributed algorithm (*HdDe*) by proposing a new mutation strategy, *GPBX-*α, and parallel execution in two separate islands using the classic DE/rand/1/bin algorithm. Wang and Zhao [27] presented a DE algorithm with a self-adaptive population resizing mechanism based on *JADE* [34], called *SapsDE*. This algorithm gradually self-adapts NP according to previous experience in generating promising solutions and enhances the performance of DE by dynamically choosing one of two mutation strategies and tuning control parameters in a self-adaptive manner. Many hybrid strategies exist that improve the efficiency of the DE algorithms. For the unconstrained global optimization problems, a novel optimization model is proposed, called clustering-based differential evolution with 2 multi-parent crossovers (*2-MPCs-CDE*) [16], hybridizing DE with the one-step k-means clustering and 2 multi-parent crossovers. Yildiz [31] developed a novel hybrid optimization algorithm called hybrid robust differential evolution (*HRDE*) by adding positive properties of Taguchis method to the DE algorithm to minimize the production cost associated with multi-pass turning problems.

Although these prominent variants of the DE algorithm largely optimize the DE process, almost all of them still suffer from problems such as difficult parameter setting, slow convergence rate, and premature convergence. Thus, this paper presents a novel adaptive DE algorithm that adopts a dynamic adjustment strategy including a directional strategy and cloud model, which we call *ADEwDC*. In the *ADEwDC* algorithm, the whole population is divided into three subgroups with each group dynamically selecting its trial vector generation from a constructed mutation and crossover pool according to its own convergence degree. The proposed algorithm realizes parameter control adaptively by referring to current fitness information. By introducing evolutionary direction [37], the convergence speed of the DE algorithm is further improved. The directional strategy is based on a greedy strategy, which means that individuals with good performance can evolve rapidly in the optimal evolution direction. These good individuals are selected according to their fitness, and the optimal direction is chosen according to the disparity between the target and trial vectors in the fitness space. However, the possibility of the algorithm falling into a local optimum is increased because of the rapid decline in population diversity. Thus, after further analysis, the *ADEwDC* algorithm improved the diversity of the population by employing the cloud model [13] in each generation. The characteristics of the cloud model, including randomness and stability, are included in the entire cloud droplet group, where randomness maintains the diversity of the population, while stability preserves the performance of excellent cloud droplets [14]. In other words, the proposed algorithm improves the convergence speed of the population, while at the same time, increasing the diversity and stability of the group. Computational experiments and comparisons show that *ADEwDC* overcomes the shortcomings of slow convergence rate and low efficiency in conventional DE algorithms, effectively avoids falling into a local optimum, and overall performs better than many state-of-the-art DE variants [4], such as *JDE* and *JaDE*, when applied to the optimization of benchmark global optimization problems.

The rest of the paper is arranged as follows. Section 2 introduces the traditional DE algorithm. In Section 3, the proposed *ADEwDC* algorithm is described in detail. The experimental design and results are presented and discussed in Section 4. Finally, the paper is concluded in Section 5.

2 The DE algorithm

Scientists and engineers from many disciplines often have to deal with the classic problems of search and optimization [4]. The DE algorithm is a simple population-based, stochastic parallel search evolutionary algorithm

for global optimization and is capable of handling non-differentiable, nonlinear, and multimodal objective functions [9, 26]. In the DE algorithm, the population consists of real-valued vectors with dimension $\Re^D$, which equals the number of design parameters. The size of the population is adjusted by parameter NP. The initial population is uniformly distributed in the search space $\left[\vec{X}_{\min}, \vec{X}_{\max}\right]$, where $\vec{X}_{\min} = \left(x_{min,1}, x_{min,2}, x_{min,3}, \ldots, x_{min,D}\right)$ and $\vec{X}_{\max} = \left(x_{\max,1}, x_{\max,2}, x_{\max,3}, \ldots, x_{\max,D}\right)$. Each component is determined as follows:

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

where $x_{i,j}$ denotes the j^{th} component of the i^{th} individual, 0 denotes the initialized subsequent generation, and *rand* is a uniformly distributed random number between 0 and 1, which is instantiated independently for each component of the i^{th} vector.

The traditional DE algorithm works through a simple cycle of stages, including mutation, crossover, and selection. Mutation and crossover are applied to each individual to produce the new population, followed by the selection phase, where each individual of the new population is compared with the corresponding individual of the old population, and the better of the two is selected as a member of the population in the next generation. A brief description of each of the evolutionary operators is given below.

Mutation In the DE literature, a parent vector from the current generation is called the *target vector*, a mutant vector obtained through the differential mutation operation is known as the *donor vector*, and finally an offspring formed by combining the donor with the target vector is called a *trial vector*. There are many mutation strategies to generate donor vector $\vec{V}_i^g$, of which the most commonly used operator and one with the simplest form is $'DE/rand/1/bin'$, which is expressed as:

$$\vec{V}_i^g = \vec{X}_{r_1^i}^g + F \cdot \left(\vec{X}_{r_2^i}^g - \vec{X}_{r_3^i}^g\right) \tag{2}$$

where $\vec{X}_{r_1^i}^g, \vec{X}_{r_2^i}^g, \vec{X}_{r_3^i}^g$ are sampled randomly from the current population in the g^{th} generation. Indices r_1^i, r_2^i, and r_3^i are mutually exclusive integers randomly chosen from the range $[1, NP]$, and which also differ from the index of the i^{th} target vector, meaning $r_1^i \neq r_2^i \neq r_3^i \neq i \in \{1, 2, 3, \ldots, NP\}$. $\vec{X}_{r_2^i}^g - \vec{X}_{r_3^i}^g$ is a differential vector, and F is a real-valued mutation scaling factor that controls the amplification of the differential variation.

Crossover After mutation, a binary crossover operation is applied to form the trial vector, $\vec{U}_i^g = \left(u_{i,1}^g, u_{i,2}^g, u_{i,3}^g, \ldots, u_{i,D}^g\right)$, to enhance the potential diversity of the population by exchanging the components of the donor vector $\vec{V}_i^g$ and target vector $\vec{X}_i^g$ according to the given probability, defined as *CR*. Each component of $\vec{U}_i^g$ is generated by the scheme outlined as:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g & if (r_j \leq CR) || j = r_i \\ x_{i,j}^g & otherwise \end{cases} \tag{3}$$

where i denotes the i^{th} individual, j denotes the j^{th} dimension, g indicates the g^{th} generation, $r_j \in [0, 1]$ is the j^{th} evaluation of a uniform random number generator. CR is the crossover constant in the range [0, 1], where zero means no crossover. $r_i \in (1, 2, 3, \ldots, D)$ is a randomly chosen index that ensures trial vector $\vec{U}_i^g$ gets at least one element from the donor vector $\vec{V}_i^g$, which is instantiated once per generation for each vector. Otherwise, no new parent vector would be produced and the population would remain unchanged. If the value of any dimension of the newly generated trial vector exceeds the pre-specified upper and lower bounds, it is set to the closest boundary value.

Selection To keep the population size constant over subsequent generations, one-to-one greedy selection between a parent and its corresponding offspring is employed to decide whether the trial individual $\vec{U}_i^g$ should replace the target vector $\vec{X}_i^g$ as a member of the next generation according to their fitness values. For minimization problems, the one-to-one selection scheme is formulated as:

$$\vec{X}_i^{g+1} = \begin{cases} \vec{U}_i^g & if \left(f\left(\vec{U}_i^g\right) \leq f\left(\vec{X}_i^g\right)\right) \\ \vec{X}_i^g & if \left(f\left(\vec{U}_i^g\right) > f\left(\vec{X}_i^g\right)\right) \end{cases} \tag{4}$$

where $f(\vec{X})$ is the objective function to be minimized. From the above description, if and only if the trial vector yields a better cost function value compared with its corresponding target vector in the current generation, it is accepted as the new parent vector in the next generation; otherwise, the target is once again retained in the population. As a result, the population either improves or remains the same in terms of fitness status, but never deteriorates.

The iterative procedure is terminated when any one of the following criteria is met: an acceptable solution is obtained, a state with no further improvement in the solution is reached, control variables have converged to a stable state, or a predefined number of iterations have been executed. Our proposed algorithm adopts a similar main framework as

the traditional *DE* algorithm, but employs different strategies in the process of evolution.

3 Adaptive differential evolutionary algorithm with directional strategy and cloud model

Based on the conventional DE algorithm, the *Adaptive Differential Evolutionary Algorithm with Directional Strategy and Cloud Model*, *ADEwDC* for short, constructs a trial vector generation pool to effect the dynamic adjustment strategy. The whole population is divided into three subgroups, with each subgroup selecting a different trial vector generation strategy from the pool according to its own *convergence degree*. In each iteration, the control parameters are dynamically set based on the fitness of the individual compared with the optimal one in the whole group, with the specific value obtained by *control parameter generation*. To improve the convergence rate, the *evolutionary direction* is introduced, and *ADEwDC* chooses the best individuals to evolve in the *optimal evolution direction*, which is defined according to the *evolution potential*. To avoid premature convergence, *ADEwDC* utilizes the *cloud model* to increase the diversity of the whole population, and proposes the *learning operator with cloud model* by applying forward and reverse cloud generators, defined as *MCG*. The specific operations of *ADEwDC* are discussed in detail in this section.

3.1 Dynamic adjustment strategy

The self-adapting strategy is an important research area for DE algorithms [36], of which there are many prominent variants. However, the performance of DE is sensitive to the choice of mutation strategy and associated control parameters [17]. In other words, different mutation strategies with different parameter settings at different stages of the evolution may be more appropriate than a single mutation strategy with unique parameter settings. Therefore, as opposed to self-adaptation [8], this paper implements dynamic adjustment by adopting a distributed topology [7] and constructing a trial vector generation pool. This means that different mutation and crossover operations, chosen from the mutation and crossover operator pool are used for each subgroup, although a uniform selection operation is adopted. Meanwhile, the control parameters for mutation and crossover obtain their values adaptively based on the fitness space, particularly from information of the marked optimal individual in the population.

Since mainly mutation and crossover in *ADEwDC* are used to obtain the optimal value, which exceeds the overall performance of the parent generation, rapidly, this paper constructs a mutation and crossover operator pool to generate the donor vector $\mathrm{P}_{\vec{V}^g} = \left(\vec{V}_1^g, \vec{V}_2^g, \vec{V}_3^g, \ldots, \vec{V}_{NP}^g\right)$ using the following variants:

1) DE/rand/1/bin

$$\mathrm{v}_{i,j}^g = \begin{cases} x_{r1,j}^g + F_1 \cdot \left(x_{r2,j}^g - x_{r3,j}^g\right) & if(r_j \le CR || n_j = j) \\ x_{i,j}^g & otherwise \end{cases} \tag{5}$$

2) DE/rand/2/bin

$$\begin{aligned} \mathrm{v}_{i,j}^g &= x_{r1,j}^g + F_1 \cdot \left(x_{r2,j}^g - x_{r3,j}^g\right) \\ &\quad + F_2 \cdot \left(x_{r4,j}^g - x_{r5,j}^g\right) \qquad if(r_j \le CR || n_j = j) \\ &= x_{i,j}^g \qquad otherwise \end{aligned} \tag{6}$$

3) DE/target-to-best/1/bin

$$\begin{aligned} \mathrm{v}_{i,j}^g &= x_{i,j}^g + F_1 \cdot \left(x_{gbest,j}^g - x_{i,j}^g\right) \\ &\quad + F_2 \cdot \left(x_{r1,j}^g - x_{r2,j}^g\right) \qquad if(r_j \le CR || n_j = j) \\ &= x_{i,j}^g \qquad otherwise \end{aligned} \tag{7}$$

4) DE/target-to-best/2/bin

$$\begin{aligned} \mathrm{v}_{i,j}^g &= x_{i,j}^g + F_1 \cdot \left(x_{gbest,j}^g - x_{i,j}^g\right) + F_2 \cdot \left(x_{r1,j}^g - x_{r2,j}^g\right) \\ &\quad + F_3 \cdot \left(x_{r3,j}^g - x_{r4,j}^g\right) \qquad if(r_j \le CR || n_j = j) \\ &= x_{i,j}^g \qquad otherwise \end{aligned} \tag{8}$$

5) DE with a neighborhood-based scheme

5.1) Neighborhood vector

$$\vec{L}_i^g = \vec{X}_i^g + \alpha_1 \cdot \left(\vec{X}_{nbest} - \vec{X}_i^g\right) + \beta_1 \cdot \left(\vec{X}_{r1}^g - \vec{X}_{r2}^g\right) \tag{9}$$

5.2) Population vector

$$\vec{G}_i^g = \vec{X}_i^g + \alpha_2 \cdot \left(\vec{X}_{gbest} - \vec{X}_i^g\right) + \beta_2 \cdot \left(\vec{X}_{r1}^g - \vec{X}_{r2}^g\right) \tag{10}$$

5.3) Component of donor vector

$$\mathrm{v}_{i,j}^g = \begin{cases} \omega \cdot g_{i,j}^g + (1-\omega) \cdot l_{i,j}^g & if(r_j \le CR || n_j = j) \\ x_{i,j}^g & otherwise \end{cases} \tag{11}$$

where $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i \in \{1, 2, 3, ..., NP\}$, j denotes the j^{th} component, F_1, F_2, F_2, α_1, α_2, β_1, β_2 are all scaling factors to control the scale of the differential vectors, ω is a weighting factor applied to the information of the neighborhood and global population, and CR is the

crossover rate to determine the source of each dimension of the offspring.

Of the five variants given above, since DE/target-to-best/1/bin and DE/target-to-best/2/bin rely on the best solution found, they usually have a faster convergence speed and perform well when solving unimodal problems. However, these algorithms are more likely to be trapped in a local optimum and to converge prematurely when solving multimodal problems. DE/rand/1/bin and DE/rand/2/bin usually demonstrate slow convergence speed with superior exploration capability. Therefore, they are usually better suited to solving multimodal problems. By including a *neighborhood-based scheme* [3, 5], different variants are employed to evolve individuals and set parameters, as discussed in the next section. Obviously, these five donor vector strategies constitute a strategy candidate pool with diverse characteristics.

In this paper, the distributed topology is simplified by dividing the population into three subgroups, while the selection criteria are strengthened by defining a variable, called the *convergence degree*.

Definition 1 Convergence degree is the average value of the accumulative fitness of each individual in the subgroup as the standard for determining the specific mutation and crossover operation. It is defined as:

$$fit^s = Fit^s_{\Sigma}/np; \tag{12}$$

where the accumulative fitness $Fit^s_{\Sigma} = \sum_{i=1}^{np} f\left(\vec{X}_i^g\right)$, s denotes the number of subgroups, and np denotes the number of individuals in each subgroup. Considering the stability of the algorithm, the subgroup with the maximum convergence degree generates the donor vector by DE/rand/1/bin or DE/rand/2/bin, the one with the minimum degree uses DE/target-to-best/1/bin or DE/target-to-best/1/bin, and the remaining ones choose their variant randomly.

ADEwDC makes full use of the historical optimal information of the population and the fitness of each individual to set the control parameters, referred to as *PSO*. The ultimate evolution goal of the current individual can be regarded as the current group optimal individual with the optimal fitness, and this forms the important theoretical basis for setting the control parameters. With reference to the relevant papers, the scaling factors F, α, β take values in $[0.1, 1]$ [19], the weighting factor ω is restricted to the range $[0.05, 0.95]$ [5], and CR takes a value in $[0.1, 0.9]$ [17]. Thus, integrating the above discussion, the dynamic adjustment strategy is given as follows.

Algorithm 1 Control parameter generation is the mechanism for generating control parameters, and is described as follows:

a. Calculate the gap between the fitness of the group optimal individual and the current individual, $\Delta_{cur-opt} = f\left(\vec{X}_{gbest}\right) - f\left(\vec{X}_i^g\right)$, with the absolute greater than zero.
b. Calculate the gap between the fitness of the individuals in the difference vector, $\Delta_{ind_1} - ind_2 = |f\left(\vec{X}_{ind_1}^g\right) - f\left(\vec{X}_{ind_2}^g\right)|$, and ensure it is positive.
c. Generate the control parameters, $\mathrm{P}_F = \Delta_{ind_1} - ind_2/\Delta_{cur-opt}$, and $\mathrm{P}_{CR} = \Delta_{cur-opt}/|f\left(\vec{X}_{gbest}\right)|$, and check the availability thereof.

Algorithm 2 Dynamic adjustment strategy

1) Construct the topology by dividing the whole population into three subgroups.
2) Calculate the convergence degree (see Definition 1) of each subgroup, and select the donor vector generation strategy from the variant pool described previously.
3) Set the control parameters by the control parameter generation algorithm (Algorithm 1), and execute the mutation and crossover operations to generate the donor vector.

At this point, *ADEwDC* generates the same number of individuals as in the parent generation, that is, the donor vector space $\mathrm{P}_{\vec{V}^g} = \left(\vec{V}_1^g, \vec{V}_2^g, \vec{V}_3^g, \ldots, \vec{V}_{NP}^g\right)$.

3.2 Design of evolutionary direction

DE belongs to the class of stochastic optimization algorithms with randomness and unpredictability, leading to the arbitrariness of the offspring and the fact that good characteristics of parental individuals cannot be fully passed to the next generation. However, living things are not passive victims of their environment in nature and human society. Instead, they struggle to fit into the environment by constantly adjusting the evolutionary direction. Therefore, this paper adopts an indicative directional propagation strategy to alter the blindness of conventional DE and improve the convergence rate of the DE algorithm before the selection process.

Definition 2 Evolutionary direction is a description of the direction vector from the current individual to its offspring for any individual $\vec{X}_i^g$ in the population P_g, such that $\forall \vec{X}_i^g \in P_g$. It is defined as:

$$\mathrm{DR}\left(\vec{X}_i^g\right) = \begin{cases} \overrightarrow{\vec{X}_i^{(g-1)'} \vec{X}_i^g} & while(g > 1) \\ rand\_direction & else \end{cases} \tag{13}$$

where $\vec{X}_i^g = offspring\left(\vec{X}_i^{(g-1)'}\right), \vec{X}_i^{(g-1)'} \in P_{(g-1)}$, and $\vec{X}_i^g \in P_g$.

In the current population, the evolutionary direction of individual $\vec{X}_i^g$ depends on itself and its corresponding parent in the previous generation of population $\vec{X}_i^{(g-1)'}$, shown in the previous equation as $\overrightarrow{\vec{X}_i^{(g-1)'} \vec{X}_i^g}$. The evolutionary directions of individuals in the first generation are selected randomly, and the optimal evolution direction is described below.

Definition 3 Optimal evolution direction $DR_{opt}(P_g)$ is selected from the evolutionary orientation of the whole population through a series of specific operations, which is the best evolution direction. It is described as:

$$DR_{opt}(P_g) = \Theta_s\left(\mathrm{DR}\left(\vec{X}_i^g\right)\right) \tag{14}$$

where $\vec{X}_i^g \in P_g$ and $f\left(\vec{X}_i^g\right) \geq f\left(\vec{X}_i^{(g-1)'}\right)$, and Θ_s denotes the selection operation on the evolution direction.

ADEwDC defines the *directional strategy* as follows: Select the top n individuals with the maximum fitness in the donor vector space as generated in the previous subsection. Select the top m optimal evolution directions. Then, evolve the selected individuals along the selected directions and shape the progeny space with $n \cdot m$ individuals, where $n \cdot m > NP$.

There are many criteria for selecting the optimal evolution direction, and the overall requirement is to improve the fitness of an individual after evolving along the selected orientation. In fact, the ultimate evolution goal of the current individual can be regarded as the current group optimal individual $\vec{X}_{gbest}$, which has the optimal fitness of the whole population, $f_{\min}^g$ for a minimization problem. Thus, the *goal vector* is defined as $\overrightarrow{\vec{X}_i^{(g-1)'} \vec{X}_{gbest}^g}$, and the *motion vector* as $\overrightarrow{\vec{X}_i^{(g-1)'} \vec{X}_i^g}$. Then, *ADEwDC* defines evolution potential as the factor used to choose the optimal directions.

Definition 4 Evolution potential ∇_{DR} is defined as the disparity between the goal vector and the motion vector in the fitness space, defined as:

$$\nabla_{\mathrm{DR}\left(\vec{X}_i^g\right)} = \left[f\left(\vec{X}_i^g\right) - f\left(\vec{X}_i^{(g-1)'}\right)\right] \Big/ \left[f\left(\vec{X}_{gbest}^g\right) - f\left(\vec{X}_i^{(g-1)'}\right)\right] \tag{15}$$

Therefore, the optimal evolution direction based on maximum evolution potential is defined as:

$$DR_{opt\_ep}(P_g) = \mathrm{DR}\left(\vec{X}_i^g\right) \quad where \quad j = \arg\max_j \nabla_{\mathrm{DR}\left(\vec{X}_i^g\right)} \tag{16}$$

The directional evolution strategy is given by the following algorithm.

Algorithm 3 Design of evolutionary direction

1) Select the top n optimal individuals from the donor vector space.
2) Calculate the evolution direction (Definition 2) using evolutionary potential (Definition 4).
3) Choose the top m directions according to the optimal evolution direction (Definition 3).
4) Evolve the selected individuals in the chosen directions and generate the trial vector.

From the optimal individuals and optimal evolution directions, the *trail vector* space $\mathrm{P}_{\vec{U}}^g = \left(\vec{U}_1^g, \vec{U}_2^g, \vec{U}_3^g, \ldots, \vec{U}_{n \cdot m}^g\right)$ is constructed, providing choices for the new generation.

3.3 Specific application of cloud model

By introducing the evolutionary direction, the convergence speed of the DE algorithm is improved further; however, the possibility of the algorithm falling into a local optimum rises markedly because of the rapid decrease in population diversity. To improve the diversity of the population, we introduce the *cloud model* [13], which describes individuals of the population through expectation *Ex*, entropy *En*, and hyper-entropy *He*.

Definition 5 Membership cloud [11] Let U denote a quantitative domain composed of precise numerical variables, with C the qualitative concept on U. If the quantitative value $x \in U$ is a random realization of qualitative concept C, the confirmation of x on C can be denoted as

$\mu(x) \in [0, 1]$, which is a random number with stable tendency.

$$\begin{aligned} &\mu : U \to [0, 1] \\ &\forall x \in U, x \to \mu(x) \end{aligned} \tag{17}$$

The distribution of x on U is called a *cloud*, x is called a *cloud droplet*, and the cloud consists of a series of cloud droplets.

The cloud droplets have a certain randomness, which maintains the diversity of individual stocks thereby avoiding a search for local extreme values, and stability characteristics, which protect the population of a more outstanding individual and thus the overall situation of extreme adaptive positioning. The membership cloud describes a concrete concept through expectation *Ex*, entropy *En*, and hyper-entropy *He*. Expectation *Ex* expresses the point that is most able to represent the domain of the concept and is the most typical sample of this concept to quantify. Entropy *En* represents the granularity of a concept that can be measured (the larger the entropy is, and the larger the granularity is, the more macro is the concept). It reflects the range of the domain space that can be accepted by the specific concept. Hyper-entropy *He* describes the uncertain measurement of entropy. It can be used to express the relationship between randomness and fuzziness.

As a specific kind of cloud, the normal cloud model has been proven to be universal [12], based on the normal distribution and Gauss membership function. Regarding probability, the normal distribution is the most commonly used form, which is described by expectation E and variance D. In fuzzy set theory, the bell-shape membership function $\mu(x) = e^{\frac{-(x-a)^2}{2b^2}}$ is also the most common membership function used in fuzzy sets. The normal cloud, described below, combines the characteristics of the two with an additional expansion.

Definition 6 Normal Cloud Model [14] Let U be the universe of discourse and $\tilde{A}$ be a qualitative concept in U. If $x \in U$ is a random instantiation of concept $\tilde{A}$, satisfying $x \sim N(E_x, En'^2)$ and $En'^2 \sim N(E_n, {H_e}^2)$, and the certainty degree of x belonging to concept $\tilde{A}$ satisfies

$$\mu = e^{-\frac{(x-E_x)^2}{2(E_n')^2}} \tag{18}$$

then the distribution of x in universe U is called a normal cloud.

Knowledge is usually the association between concepts in the real world, between which the cause and effect relationship can be described by the membership cloud generator (*MCG*), including both a *forward* and *reverse* generator (for more information, see [11]). With the associated numerical characteristics, that is, *Ex*, *En* and *He*, the forward cloud generator can generate cloud drops (x, μ), where x is the quantity values and μ is the membership degree of x [10]. The reverse cloud generator is the other conversion model that can convert quantity numbers to a quality concept. It can convert accurate data $(x_1, x_2, ..., x_n)$ with membership degrees $(\mu_1, \mu_2, ..., \mu_n)$ to a quality cloud concept expressed as numerical characteristics (E_x, E_n, H_e) [10]. The *MCG* based on cloud theory is summarized below.

Algorithm 4 MCG

1. Forward normal cloud generator $f_{CG}(E_x, E_n, H_e, N)$ [14]

   Input: Three parameters E_x, E_n, H_e and the number of required cloud drops, N
   Output: N cloud drops and their certainty degree
   STEP Execute the following steps:

   Step 1 Generate a normally distributed random number ${E_{n_i}}'$ with expectation E_n and variance ${H_e}^2$ that satisfies ${E_{n_i}}' \sim N(E_n, {H_e}^2)$.
   Step 2 Generate a normally distributed random number x_i with expectation E_x and variance ${E_{n_i}}'$ that satisfies $x_i \sim N(E_x, ({E_{n_i}}')^2)$.
   Step 3 Calculate $\mu_i = e^{-\frac{(x_i-E_x)^2}{2({E_{n_i}}')^2}}$.
   Step 4 x_i is a cloud drop in the universe and μ_i is the certainty degree that it belongs to concept $\tilde{A}$.
   Step 5 Repeat steps 1−4 until N cloud drops have been generated.

2. Reverse normal cloud generator $r_{CG}(x_1, x_2, x_3, ..., x_n)$ [39]

   Input: Accurate data $(x_1, x_2, x_3, ..., x_n)$
   Output: The numerical characteristic (E_x, E_n, H_e)
   STEP Execute the following steps:

   1) Calculate expectation, $E_x = Mean(x_1, x_2, x_3, ..., x_n)$.
   2) Calculate entropy, $E_n = Stdev(x_1, x_2, x_3, ..., x_n)$.
   3) Calculate ${E_{n_i}}' = \sqrt{-(x_i - E_x)^2 / 2ln\mu_i}$.
   4) Calculate hyper-entropy, $H_e = Stdev({E_{n_1}}', {E_{n_2}}', {E_{n_3}}', ..., {E_{n_n}}')$.

Based on our analysis, *ADEwDC* incorporates the *cloud model* into the selection operation to remedy the diversity of

population, and presents a novel operator called the *learning operator with cloud model*.

Definition 7 Learning operator with cloud model View the current population I^{λ} as a cloud defined using $Cloud_M(E_x, E_n, H_e)$, the eigenvalue calculated by the reverse normal cloud generator $r_{CG}(P_\lambda)$ to describe the overall information owned by it. For each individual in the suboptimal population, $\forall d_i \in I^{\lambda-e}$, generate a new individual using the forward normal cloud generator $f_{CG}(E_x, E_n, H_e, (\lambda - e))$, then the learning operator with cloud model is defined as:

$$\forall d_i \in I^{\lambda-e}, \exists d_c \in \Upsilon_{Cloud}^{\lambda-e} : d_i \rightarrow d_c,$$

$$\Upsilon_{Cloud}^{\lambda-e} = fCG(E_x, E_n, H_e, (\lambda - e)) \tag{19}$$

where λ denotes the parents' individuals space and e means the individuals selected from trail vector space. That is, part of the subgeneration is selected directly from the *trial vector space*, with count e, and the rest is generated randomly by the *learning operator with cloud model* according to the feature information about the known population.

The specific application of the cloud model is summarized below.

Algorithm 5 Specific application of cloud model

1) begin
2) Provide a known population with each parameter vector, $P_\lambda = \left\{ \overrightarrow{X}_1^g, \overrightarrow{X}_2^g, \overrightarrow{X}_3^g, \ldots, \overrightarrow{X}_\lambda^g \right\}$.
3) Obtain the numerical characteristics of the population using the reverse cloud generator, $Cloud_M(E_x, E_n, H_e) = r_{CG}(P_\lambda)$.
4) Confirm the number of droplets that need to be generated randomly, $(\lambda - e)$.
5) Generate the known number of individuals using the forward cloud generator, $\Upsilon_{Cloud}^{\lambda-e} = f_{CG}(E_x, E_n, H_e, (\lambda - e))$.
6) end

Thus far, we have introduced each part of *ADEwDC*: the *dynamic adjustment strategy* is used in the mutation and crossover process, while the *design of the evolutionary direction* and *specific application of the cloud model* are both applied in the selection process. The overall framework of *ADEwDC* is presented next.

3.4 ADEwDC

The proposed *ADEwDC* employs the dynamic adjustment strategy (see Section 3.1), the directional strategy (see Section 3.2), and the cloud model (see Section 3.3), introduced previously. In this section, we first present the overall framework of the proposed algorithm, and then analyze its discipline holistically.

3.4.1 Framework of the algorithm
3.4.2 Discipline of the algorithm

The main framework of *ADEwDC* is taken from the traditional DE algorithm, but with changed mutation and crossover strategies and an improved selection mechanism. As with the classical *DE* algorithm, the proposed algorithm defines a *donor vector* and a *trial vector*, where the former is the result of the mutation and crossover operation, while the latter is the result of directional evolution. The next generation may come from the parent generation, donor vector space (Algorithm 2), or trial vector space (Algorithm 3), but most new individuals must be generated by the cloud model (Algorithm 5).

a) ***Dynamic adjustment strategy*** *ADEwDC* constructs a mutation and crossover pool to realize the dynamic adjustment strategy, including DE/rand/1/bin, DE/rand/2/bin, DE/target-to-best/1/bin, DE/target-to-best/2/bin, and DE with neighborhood-based scheme variants. It then defines the strategies to select the evolution operators (Definition 1) and set the control parameters (Algorithm 1) based on the constructed topology, which are summarized as the *donor vector generation* (Algorithm 2).
b) ***Design of evolutionary direction*** The evolutionary direction is determined based on the theory that the individuals are not passive in joining the environment, but struggle to fit into the environment by constantly adjusting the direction of evolution. This algorithm defines the optimal direction (Definition 3) with evolution potential (Definition 4), and synthesizes the *trial vector generation*(Algorithm 3).
c) ***Specific application of cloud model*** The cloud model is a useful mathematical tool with randomness and a stability tendency, which is employed to record the eigenvalues of the parent generation using the *reverse cloud generator* and generate cloud droplets randomly for most of the filial generation using the *forward cloud generator* (Definition 4). The specific operation is described by Algorithm 5.
d) ***Selection strategy*** The next generation consists of two parts: some are directly selected as offspring from the vector space, consisting of the parent population, the donor vector space, and the trial vector space, whereas

Algorithm 6 ADEwDC

1) Initialize: Set the generation number $g = 0$ and randomly initialize a population of NP individuals $P_g = \left\{\vec{X}_1^g, \vec{X}_2^g, \vec{X}_3^g, \ldots, \vec{X}_{NP}^g\right\}$ with $\vec{X}_1^g = \left(x_{i,1}^g, x_{i,2}^g, x_{i,3}^g, ..., x_{i,D}^g\right)$ and each individual uniformly distributed in the range $\left[\vec{X}_{\min}, \vec{X}_{\max}\right]$, where $\vec{X}_{\min} = (x_{min,1}, x_{min,2}, x_{min,3}, ..., x_{min,D})$ and $\vec{X}_{\max} = (x_{\max,1}, x_{\max,2}, x_{\max,3}, ..., x_{\max,D})$ with $i = 1, 2, 3, ..., NP$.
2) **WHILE** the termination criterion is not satisfied **DO**
   2.1) Evaluate the fitness space: $\text{Fit}_g = \left(f\left(\vec{X}_1^g\right), f\left(\vec{X}_2^g\right), f\left(\vec{X}_3^g\right), \ldots f\left(\vec{X}_{NP}^g\right)\right)$ where Fit_g denotes the current fitness space, and obtain the current group optimal individual $\vec{X}_{gbest}^g$ and its fitness $Fit_{gbest} = f(\vec{X}_{gbest}^g)$.
   2.2) Donor vector generation: Employ the dynamic adjustment strategy (Algorithm 2) to choose the mutation and crossover operation (Definition 1) and set the control parameters (Algorithm 1), then generate the donor vector space $P_{\underset{V}{\rightarrow g}} = \left(\vec{V}_1^g, \vec{V}_2^g, \vec{V}_3^g, \ldots, \vec{V}_{NP}^g\right)$ with the same scale as the parent generation.
   2.3) Trial vector generation: Adopt the directional strategy (Algorithm 3) to evolve the outstanding individuals along the optimal evolution direction (Definition 3) with great evolution potential (Definition 4). Then, generate the trial vector space $P_{\underset{U}{\rightarrow g}} = \left(\vec{U}_1^g, \vec{U}_2^g, \vec{U}_3^g, \ldots, \vec{U}_{n \cdot m}^g\right)$, where n denotes the top n individuals in the donor vector space $P_{\underset{V}{\rightarrow g}}$ and m denotes the top m optimal evolution directions between the parent vector $\vec{X}_j^g$ and its corresponding donor vector $\vec{V}_j^g$.
   2.4) Selection strategy:
      2.4.a Directly select a certain number of the best individuals (φ^e) from the $P_g + P_{\underset{V}{\rightarrow g}} + P_{\underset{U}{\rightarrow g}}$ vector space as one part of the subgeneration, where e denotes the number of these outstanding individuals through which the excellent performance of the population can be preserved.
      2.4.b Utilize the cloud model (Algorithm 5 and Definition 7) to generate the rest of the subgeneration $\Upsilon_{Cloud}^{\lambda-e}$ that replaces the corresponding pre-generation $I^{\lambda-e}$, thereby effectively maintaining the diversity of the population and avoiding falling into a local optimum.
      2.4.c Reconstruct the population by assembling the above two results (φ^e and $\Upsilon_{Cloud}^{\lambda-e}$) as the next generation or subgeneration P_{g+1}, given as $P_{g+1} = \varphi^e \cup \Upsilon_{Cloud}^{\lambda-e}$.
   2.5) Increase the generation count: $g = g + 1$.
3) **END WHILE**

the greater part is generated by the *MCG* (Algorithm 4) based on the cloud model to compensate for the loss of population diversity, and which is described as:

$$P_{g+1} = \Theta_s^e \left(P_g \cup P_{\underset{V}{\rightarrow g}} \cup P_{\underset{U}{\rightarrow g}}\right) \cup \Upsilon_{Cloud}^{\lambda-e} \tag{20}$$

where Θ_s^e means selecting e individuals from the specific vector space.

ADEwDC utilizes the dynamic adjustment strategy to select evolutionary variants and set control parameters based on the constructed topology. In fact, the subgroup with the maximum fit^s evolves with a slow convergence speed and superior exploration capability, while the minimum one evolves with a fast convergence speed and superior exploitation capability. Furthermore, each individual in the same subgroup sets its control parameters based on the gap between its fitness and the best fitness of the population. In short, this algorithm has good adaptability.

Furthermore, *ADEwDC* designs an evolutionary direction to improve the convergence rate, introduces the cloud model to ensure good diversity of the population, and applies the special selection strategy to balance the performance of the whole group. A series of experiments were carried out to confirm the effectiveness of *ADEwDC* as reported in the next section.

4 Experimental setup and results

To validate the *ADEwDC* algorithm, we selected several categories of global minimization benchmark functions to evaluate the proposed algorithm against other DE variants. These benchmark functions provide a balance between unimodal and multimodal functions, and were chosen from the set of 13 classic benchmark problems [30] that have frequently been used in the literature [25, 38]. The following problems were used:

(1) *Rastrigin function*:

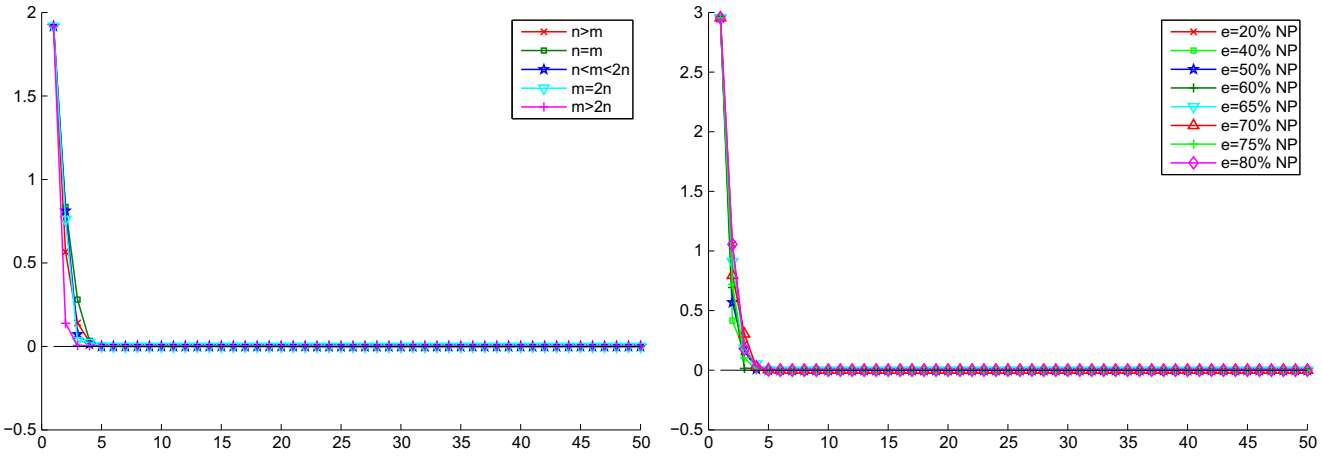
$$f_1(X) = 10 \cdot D + \sum_{i=1}^{D} \left[x_i^2 - 10 \cdot cos(2\Pi x_i)\right]$$

with global optimum $X^* = 0$ and $f(X^*) = 0$ for $-5 \le x_i \le 5$.

(2) *Sphere function* :

$$f_2(X) = \sum_{i=1}^{D} x_i^2$$

with global optimum $X^* = 0$ and $f(X^*) = 0$ for $-100 \le x_i \le 100$.



(a) Effect of varying n and m with the same e (b) Effect of varying e with the same n and m

Fig. 1 Influence of the number of top individuals (n), the number of top directions (m), and the number of remaining individuals (e) when optimizing the Rastrigin function with *ADEwDC*

(3) *Rosenbrock function* :

$$f_3(X) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$$

with global optimum $X^* = (1, 1, ..., 1)$ and $f(X^*) = 0$ for $-100 \leq x_i \leq 100$.

(4) *Ackley function* :

$$f_4(X) = -20 \cdot e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\Pi x_i)} + 20 + e$$

with global optimum $X^* = 0$ and $f(X^*) = 0$ for $-32 \leq x_i \leq 32$.

(5) *Griewank function* :

$$f_5(X) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

with global optimum $X^* = 0$ and $f(X^*) = 0$ for $-600 \leq x_i \leq 600$.

Experiment 1 used a two-dimensional (2D) f_1 to explore the impact of the relevant parameters of Algorithm 6, while experiment 2 used the same test function to show the distribution of individuals in the solution space of each generation and verify its convergence. Experiment 3 used a 2D f_2, f_3, f_4, f_5 to explain the effectiveness of *ADEwDC*, compared with the conventional DE algorithm, DE/rand/1/bin. Next, *ADEwDCf* was evaluated on the CEC2013 benchmark problem set [15], as well as the CEC2005 benchmarks [23] in Experiment 4. In this paper, we compare *ADEwDC* with the following state-of-the-art DE algorithms: *SHADE* [24], *CoDE* [28], *EPSDE* [17], *JADE* [34], and *dynNP-jDE* [1] (an improved version of *jDE* [2]). Our comparisons were done with dimensions $D = 30$ to analyze the effectiveness of the *ADEwDC* algorithm comprehensively. Finally, *ADEwDC* was used to solve the CEC2013 benchmark problems with dimensions $D = 50$ to prove its universality and robustness.

All experiments were executed on the following system:

- OS: Windows 7 Professional
- CPU: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz 2.40GHz
- RAM: 12.0GB
- Language: Matlab
- Compiler: Microsoft Visual C++ 2012

4.1 Experiment 1 - related parameter settings

To investigate the relevant parameter settings, we utilized a 2D f_1 as the test function to explore the influence

Table 1 Average best solution of all the experimental results varying n and m with a fixed e when optimizing the Rastrigin function with *ADEwDC* over 10000 trials

Case	$n > m$	$n = m$	$n < m < 2n$	$m = 2n$	$m > 2n$
Avg. value	1.81e−07	2.43e−07	1.99e−04	1.97e−04	**1.11e−08**

Table 2 Average best solution of all the experimental results varying ***e*** with fixed values of ***n*** and ***m*** when optimizing the Rastrigin function with *ADEwDC* over 10000 trials

e	20 %NP	40 %NP	50 %NP	60 %NP	65 %NP	70 %NP	75 %NP	80 %NP
Avg. value	1.63−03	1.14e−04	3.15e−04	**1.70e−08**	1.36e−03	2.69e−03	4.27e−03	5.41e−03

of parameters in Algorithm 6, with the function described as:

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

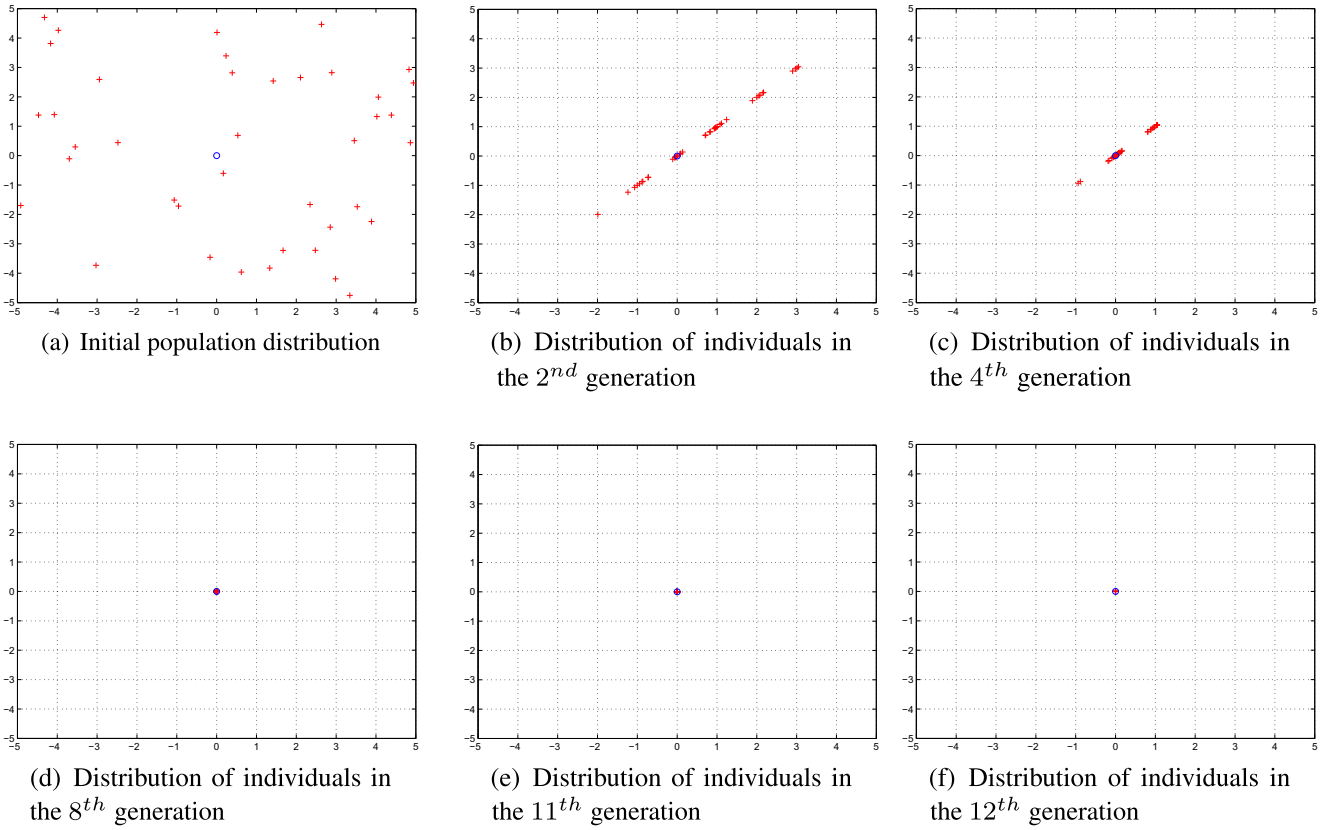
There are many local minima in the region of the value distribution of this function, and thus it is a good test case for measuring the effect of the relevant parameter settings. In this experiment, the number of top individuals (***n***) and the number of top directions (***m***) in step 2.3 of Algorithm 6 were observed, as well as the number of remaining individuals (***e***) from step 2.4 of Algorithm 6.

First, keeping the same ***e***, we set $n > m$, $n = m$, $n < m < 2n$, $m = 2n$ and $m > 2n$ as five cases. This experiment was repeated 10,000 times, and we randomly selected the set of experimental results shown in Fig. 1a. Then we calculate the average best solution of all the experimental results, which are shown in Table 1.

Then, keeping the same ***n*** and ***m***, set $e = 20\ \%\ NP$, $e = 40\ \%\ NP$, $e = 50\ \%\ NP$, $e = 60\ \%\ NP$, $e = 65\ \%\ NP$, $e = 70\ \%\ NP$, $e = 75\ \%\ NP$ and $e = 80\ \%\ NP$ as eight cases. This experiment was also repeated 10,000 times, and we randomly selected the set of experimental results shown in Fig. 1b, and then calculate the average best solution of all the experimental results in Table 2.

Based on the above analysis, *ADEwDC* achieves relatively good results when the number of top directions (***m***) is more than twice the number of top individuals (***n***), and the number of remaining individuals (***e***) is **60** % of the population size. In fact, ***n*** satisfies:

$$\begin{cases} m > 2n \\ n \cdot m > NP \end{cases} \Rightarrow n \cdot m > 2n^2 > NP \Rightarrow n > \sqrt{\frac{1}{2} \cdot NP} \tag{21}$$



(a) Initial population distribution

(b) Distribution of individuals in the 2^{nd} generation

(c) Distribution of individuals in the 4^{th} generation

(d) Distribution of individuals in the 8^{th} generation

(e) Distribution of individuals in the 11^{th} generation

(f) Distribution of individuals in the 12^{th} generation

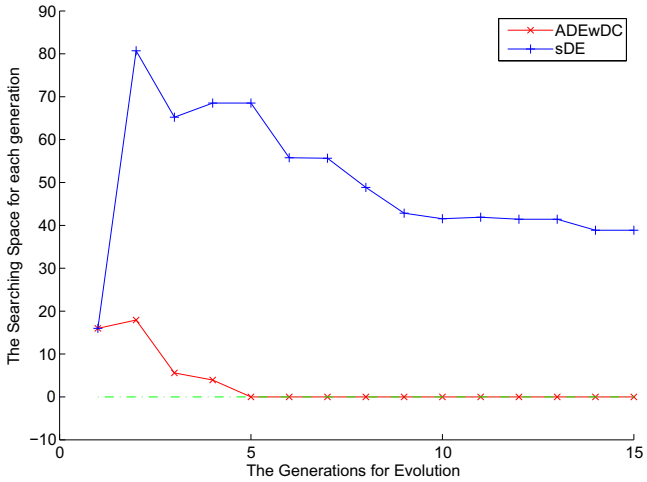
Fig. 2 Distribution of individuals in each generation while computing the minimum of the Rastrigin function

4.2 Experiment 2 - convergence analysis

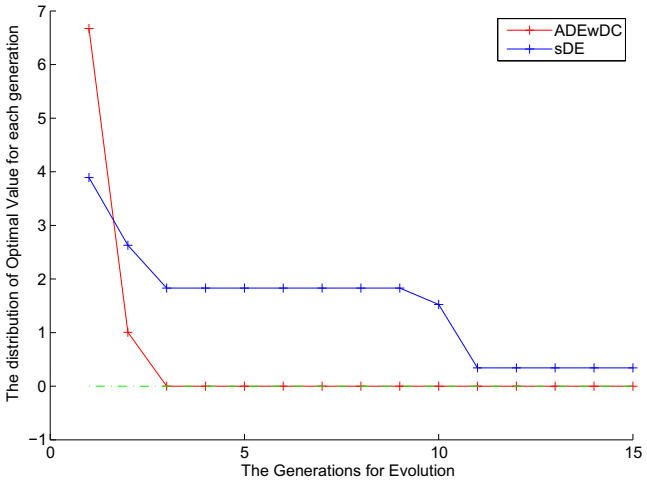
As defined above, the 2D Rastrigin function was also used in this experiment, although it was applied to verify the convergence of Algorithm 6.

The initial population consisted of 40 points, 20 times the dimension recommended by Price [20], distribution randomly in the region between $[-5, 5]$ and $[-5, 5]$. Meanwhile, $n = 5, m = 12$, ensure $n \cdot m > NP$ and $m > 2n$, then $e = 24$, which is 60 % of the population size. The distribution of the initial population is shown in Fig. 1a. The fitness function is the function value of the concrete points in the area. From the test results shown in Fig. 2a–f, we can see that after the evolution of 12 generations, the individuals distribution converges well. We listed the individuals of the 2^{nd}, 4^{th}, 8^{th}, 11^{th}, 12^{th} generations on the region as shown in Fig. 2b–f, respectively.

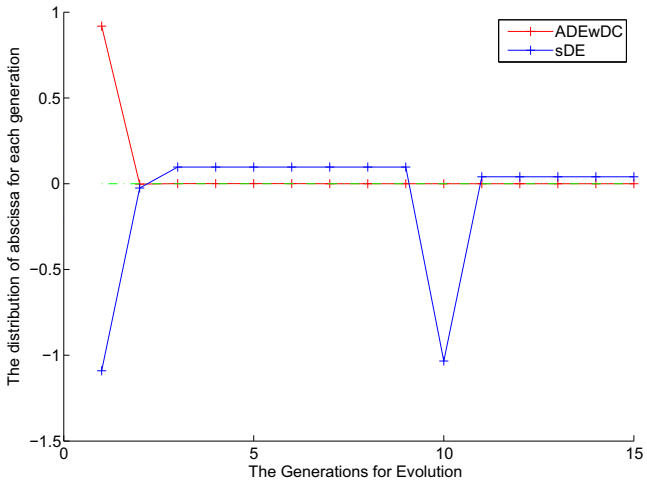
To further verify the convergence of *ADEwDC*, it was compared with the standard DE, *sDE* for short, with DE/rand/1/bin (with $F = 0.5, CR = 0.3$ in our experiments). The results of evolving 15 generations shown in Fig. 3 were randomly selected from 50 trial runs. In the search procedure for the optimal solution, we used the distribution area of all the points to represent the search space, changes in which are illustrated in Fig. 3a. Obviously, the search space of *ADEwDC* tends to zero much faster than that of *sDE*, after evolving only five or six generations. We plot the best individual for each evolutionary generation, as well as its abscissa and ordinates in Fig. 3b–d, respectively. In Fig. 3, it can be seen that



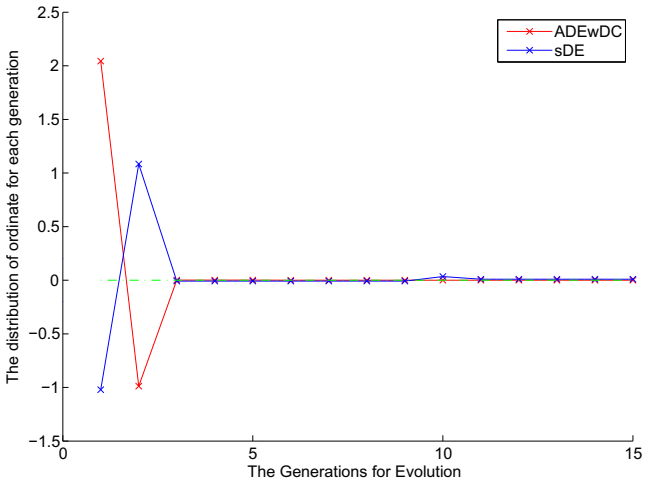
(a) Search area for each generation



(b) Optimal individual in each generation



(c) Abscissa of the optimal individual in each generation



(d) Ordinate of the optimal individual in each generation

Fig. 3 Changes in the search space and the optimal value in each generation, comparing ADEwDC with the standard DE

ADEwDC converges to the optimal solution much faster than *sDE*.

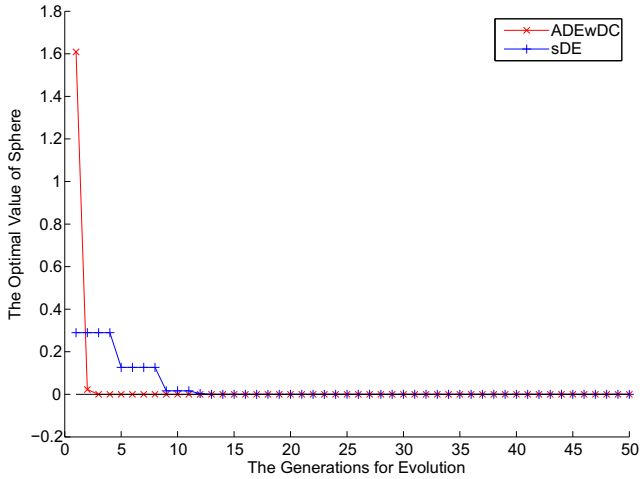
Overall, the results of experiment 2 show that *ADEwDC* can effectively reduce the search space and rapidly converge to the optimal solution, which is consistent with the directional strategy. In the course of evolution, excellent individuals evolve along the optimal evolution directions, enabling the entire population to obtain the optimal value easily and quickly.

4.3 Experiment 3 - effectiveness analysis

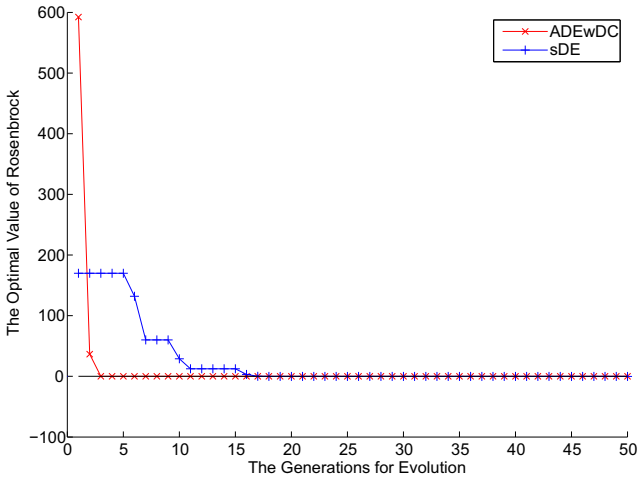
In this experiment, *ADEwDC* was used to optimize the 2D f_2, f_3, f_4, f_5 to investigate the effectiveness of the algorithm compared with the standard DE algorithm (*sDE*). The results of evolving 50 generations for each function, shown in Fig. 4, were randomly selected from 100 trial runs.

For the Ackley and Griewank functions, the results by *ADEwDC* are better than those by *sDE* in terms of both convergence speed and accuracy, as clearly shown in Figs. 4c and (d). For the Sphere and Rosenbrock functions, as shown in Fig. 4a and b, *ADEwDC* converges faster than *sDE* and obtains the desired optimal solution. To illustrate the effectiveness of *ADEwDC* further, in Table 3 we list one full trial result, randomly selected from the 100 experimental results for each function.

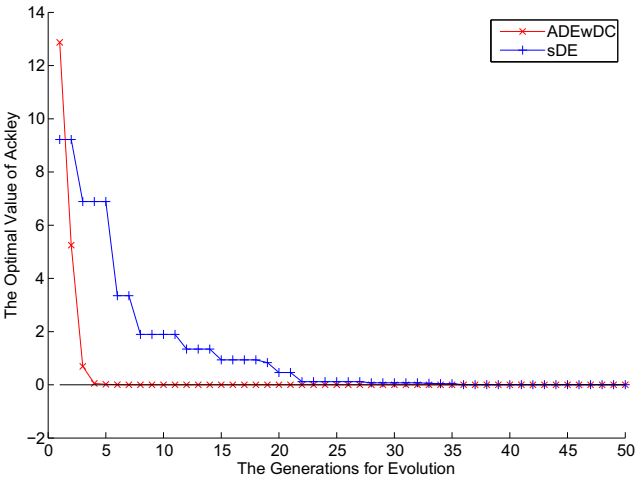
From all the test results, we find that *ADEwDC* converges to the optimal solution quickly and has good robustness. In contrast, *sDE* is more likely to fall into a local optimum



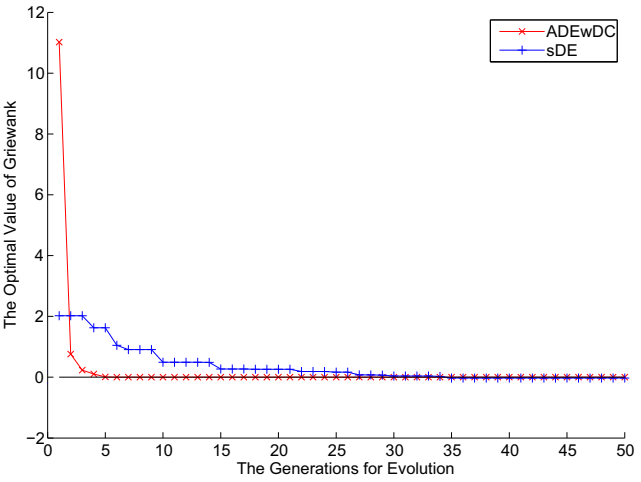
(a) Comparison when computing the minimum of the Sphere function



(b) Comparison when computing the minimum of the Rosenbrock function



(c) Comparison when computing the minimum of the Ackley function



(d) Comparison when computing the minimum of the Griewank function

Fig. 4 Comparison of the standard DE and the proposed algorithms while computing the minimum of f_2, f_3, f_4, f_5

Table 3 Comparison of the standard DE algorithm and *ADEwDC* when computing the minimum of f_2, f_3, f_4, f_5

Function	Algorithm	1*st*	2*nd*	4*th*	8*th*	12*th*
Sphere	ADEwDC	16.5860	1.7209	0.0223	$3.9371e-08$	$2.4060e-13$
	sDE	16.5860	16.5860	16.5860	2.2757	2.2757
Rosenbrock	ADEwDC	$1.2202e+06$	134.2504	1.4721	0.0040	$1.9321e-06$
	sDE	$1.2202e+06$	$4.7175e+04$	$4.7175e+04$	14.8713	14.8713
Ackley	ADEwDC	16.6727	4.0396	0.1190	$1.3502e-04$	$1.7969e-07$
	sDE	16.6727	10.3736	9.4391	8.3671	3.5868
Griewank	ADEwDC	5.4584	1.0759	0.0464	$3.7112e-04$	$3.7112e-04$
	sDE	5.4584	4.1087	3.3114	0.8975	0.5499
Function	Algorithm	16*th*	20*th*	30*th*	40*th*	50*th*
Sphere	ADEwDC	$1.8366e-19$	$3.7332e-27$	$2.5965e-40$	$4.1011e-55$	$4.8661e-69$
	sDE	0.9714	0.6566	$5.3771e-04$	$1.0916e-04$	$1.6398e-06$
Rosenbrock	ADEwDC	$2.6971e-07$	$5.1569e-09$	$1.3818e-11$	$2.7095e-14$	$9.4291e-19$
	sDE	6.1316	6.1316	6.1316	3.7630	0.8860
Ackley	ADEwDC	$2.5507e-10$	$5.5511e-13$	$8.8818e-16$	$8.8818e-16$	$8.8818e-16$
	sDE	3.5868	1.9990	0.3055	0.0027	0.0027
Griewank	ADEwDC	$5.6048e-10$	$4.9405e-14$	0	0	0
	sDE	0.3601	0.0719	0.0719	0.0119	0.0119

as shown in Fig. 5. Moreover, 1.0×10^{-3} is defined as the acceptable level, which means that the run is judged to be successful if a solution obtained by an algorithm falls between the acceptable level and the actual global optimum [33]. And we compare the frequency of premature convergence and the average value of all the test results between *ADEwDC* and *sDE* in Table 4.

Generally, the payoff for increasing diversity is a slower (although more efficient) convergence speed. Nevertheless, Table 4 indicates that *ADEwDC* is not at the cost of the convergence speed to obtain the diversity of the solutions; conversely, it improves the diversity with **MCG** (Algorithm 4), maintaining a good convergence rate at the same time which has been confirmed in the previous content [6, 35].

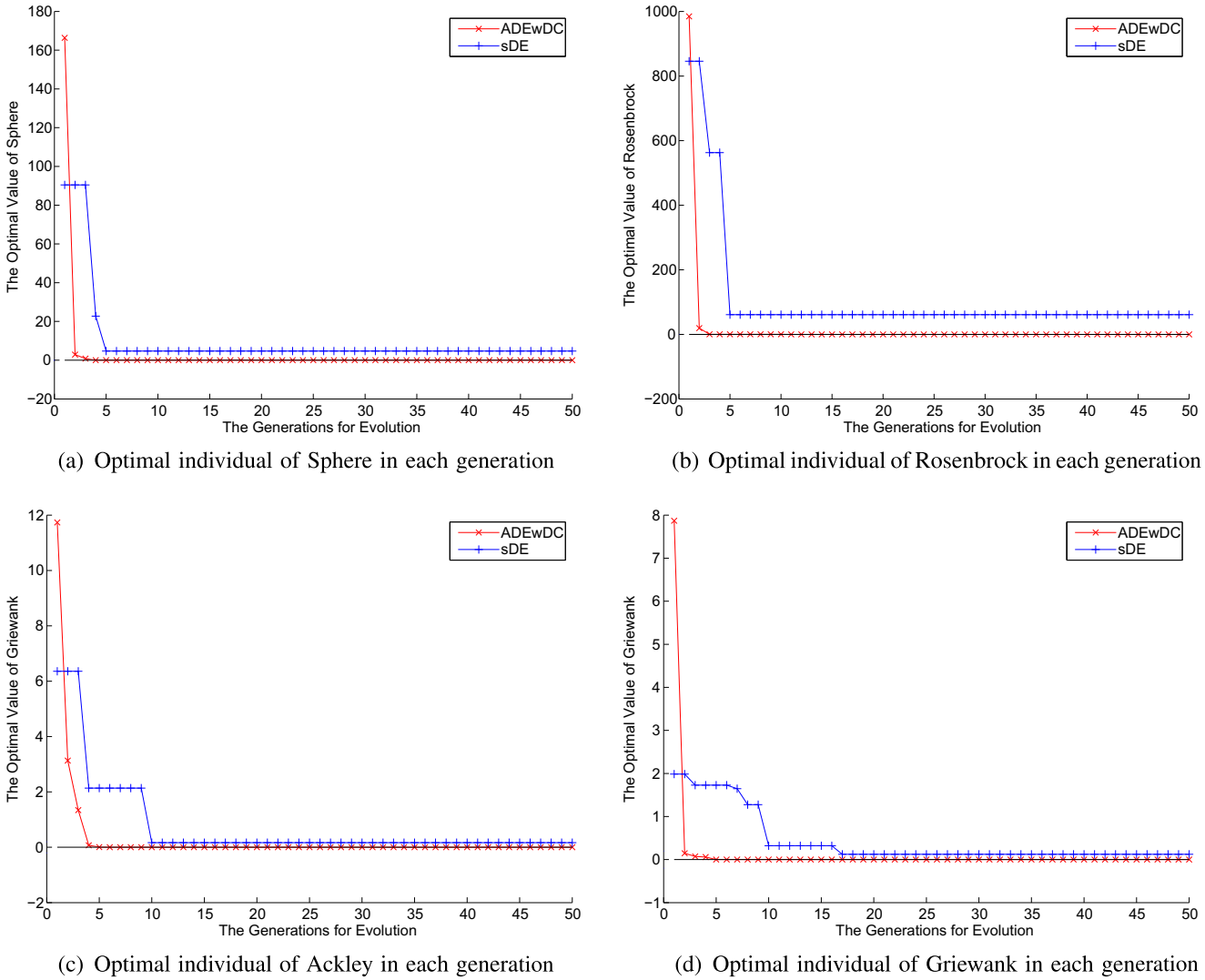
From the above experimental results and analysis, it can been seen that compared with the standard DE algorithm, Algorithm 6 has a faster convergence rate, obtains more accurate optimization results, and has better robustness. In short, *ADEwDC* is more effective than the standard DE because of the cloud model, which enhances the population diversity and prevents the algorithm from falling into a local optimum.

4.4 Experiment 4 - competitiveness analysis

In this section, we evaluate the performance of *ADEwDC* on the CEC2013 benchmark problem set [15], compared with *SHADE* [24], *CoDE* [28], *EPSDE* [17], *JADE* [34] and *dynNP-jDE* [1]. Then, using the CEC2005 benchmarks [23], *ADEwDC* is compared with SHADE to verify its validity and accuracy further. For each comparative algorithm, we used the control parameter values suggested in the cited papers. *ADEwDC* was executed on the same system as that given for the previous experiments, while comparative data for *SHADE* were taken from its original paper [24]. The source programs for CoDE, EPSDE, and JADE were based on code received from the original authors. The number of dimensions was set to $D = 30$ and the maximum number of objective function calls per run was calculated as $D \times 10,000$ (that is, 300,000) when comparing *ADEwDC* with the other algorithms. Meanwhile, $NP = 100, n = 8, m = 20$, and $e = 60$. Finally, we executed *ADEwDC* on the CEC2013 benchmarks with the number of dimensions set to $D = 50$ as a high-dimensional case to prove its universality and robustness.

4.4.1 Experiment on the CEC2013 benchmarks

In this experiment, we performed our evaluation following the guidelines of the CEC2013 benchmark competition [15]. The search space was set as $[-100, 100]^D$ for all the selected problems, the results of which are shown in Table 5. In the table, the mean and standard deviation of the error (difference) between the best fitness values found in each run and the optimal value are shown. The $+$, $-$, $\approx$ indicate whether a given algorithm performed significantly better $(+)$, significantly worse $(-)$, or not significantly



(a) Optimal individual of Sphere in each generation

(b) Optimal individual of Rosenbrock in each generation

(c) Optimal individual of Ackley in each generation

(d) Optimal individual of Griewank in each generation

Fig. 5 Optimal individuals of f_2, f_3, f_4, f_5 in each generation computed by *ADEwDC* and *sDE*, showing that ADEwDC has better robustness than sDE

different better or worse ($\approx$) compared to *ADEwDC* according to the Wilcoxon rank-sum test, which is a nonparametric alternative to the two-sample t-test based solely on the order of the observations from the two samples (significance threshold $p \leq 0.05$) [29]. Functions $f_1 \sim f_5$ are unimodal, while $f_6 \sim f_{20}$ are multimodal. $f_{21} \sim f_{28}$ are composite functions combining multiple test problems into a complex landscape [15].

The best result for each benchmark function is highlighted in Table 5, while Table 6 shows the statistical ranking according to the respective performance of the DE algorithms. The algorithms are ranked based on the average best solution in each row of Table 5, and the final rank of the algorithms is determined by the average rank calculated according to problem feature in each column. From Table 6, we see that *SHADE*, *ADEwDC* and *JADE* achieve

Table 4 The frequency of premature convergence and the average best solution when computing the minimum of f_2, f_3, f_4, f_5 with *ADEwDC* and *sDE*

Algorithm		Sphere	Rosenbrock	Ackley	Griewank
ADEwDC	Frequency	0	0	0	0
	Average	1.73e−30	3.14e−24	8.86e−16	0.00e+00
sDE	Frequency	97	99	99	100
	Average	4.37e−03	2.09e−02	5.94e−02	1.33e−02

Table 5 Comparison of *ADEwDC* and state-of-the-art DE algorithms while computing the minimum of the benchmark functions in CEC2013. For all problems, dimensionality was set to $D = 30$, and the maximum number of objective function evaluations to $D \times 10,000 = 300,000$. All results are averaged over 51 trials

F	ADEwDC Mean(Std)	SHADE Mean(Std)	CoDE Mean(Std)	EPSDE Mean(Std)	JADE Mean(Std)	dynNP-jDE Mean(Std)
f_1	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈
f_2	**1.89e+03(3.64e+03)**	9.00e+03(7.47e+03)−	9.78e+04(4.81e+04)−	1.37e+06(5.39e+06)−	7.67e+03(5.66e+03)−	9.52e+04(4.09e+04)−
f_3	1.80e+06(2.36e+06)	**4.02e+01(2.13e+02)**+	1.08e+06(3.03e+06)+	1.75e+08(5.39e+08)−	4.71e+05(2.35e+06)+	1.71e+06(2.54e+06)≈
f_4	1.00e−03(2.30e−02)	**1.92e−04(3.01e−04)**+	8.18e−02(1.09e−01)−	8.08e+03(2.56e+04)−	6.09e+03(1.33e+04)−	4.76e+01(4.75e+01)−
f_5	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈
f_6	**0.00e+00(0.00e+00)**	5.96e−01(3.73e+00)−	4.16e+00(9.00e+00)−	9.27e+00(1.33e+00)−	2.07e+00(7.17e+00)−	1.19e+01(1.66e+00)−
f_7	5.57e+01(1.24e+01)	4.60e+00(5.39e+00)+	9.32e+00(6.37e+00)+	5.88e+01(4.29e+01)−	3.16e+00(4.13e+00)+	**2.62e+00(1.59e+00)**+
f_8	2.08e+01(0.00e+00)	**2.07e+01(1.76e−01)**+	2.08e+01(1.18e−01)≈	2.09e+01(5.32e−02)−	2.09e+01(4.93e−02)−	2.10e+01(3.98e−02)−
f_9	2.61e+01(3.89e+00)	2.75e+01(1.77e+00)−	**1.45e+01(2.90e+00)**+	3.50e+01(4.21e+00)−	2.65e+01(1.96e+00)−	2.20e+01(5.12e+00)+
f_{10}	5.41e−02(6.87e−02)	7.69e−02(3.58e−02)−	**2.71e−02(1.50e−02)**+	1.02e−01(5.65e−02)−	4.04e−02(2.37e−02)+	3.63e−02(2.34e−02)+
f_{11}	**3.89e−02(8.33e−01)**	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈	**1.95e−02(1.39e−01)**≈	**0.00e+00(0.00e+00)**≈	**0.00e+00(0.00e+00)**≈
f_{12}	4.77e+01(1.44e+01)	2.30e+01(3.73e+00)−	3.98e+01(1.21e+01)≈	4.94e+01(9.28e+00)−	**2.29e+01(5.45e+00)**+	4.07e+01(8.81e+00)−
f_{13}	8.59e+01(2.86e+01)	5.03e+01(1.34e+01)+	8.04e+01(2.74e+01)≈	7.68e+01(1.72e+01)+	**4.67e+01(1.37e+01)**+	7.10e+01(1.72e+01)+
f_{14}	2.59e−01(6.87e−01)	3.18e−02(2.33e−02)+	3.60e+00(4.09e+00)−	3.99e−01(6.00e−01)−	2.86e−02(2.53e−02)+	**9.39e−03(1.40e−02)**+
f_{15}	**3.19e+03(6.11e+02)**	3.22e+03(2.64e+02)−	3.36e+03(5.31e+02)−	6.75e+03(7.60e+02)−	3.24e+03(3.17e+02)−	4.39e+03(4.72e+02)−
f_{16}	**4.74e−01(0.00e+00)**	9.13e−01(1.85e−01)−	3.38e−01(2.03e−01)≈	2.48e+00(2.88e−01)−	1.84e+00(6.27e−01)−	2.32e+00(2.83e−01)−
f_{17}	3.04e+01(1.33e−02)	**3.04e+01(3.83e−14)**≈	3.04e+01(1.17e−02)−	3.04e+01(2.51e−02)≈	3.04e+01(1.95e−14)≈	3.04e+01(1.78e−03)≈
f_{18}	8.06e+01(0.00e+00)	7.25e+01(5.58e+00)+	**6.69e+01(9.23e+00)**+	1.37e+02(1.12e+01)−	7.76e+01(5.91e+00)−	1.35e+02(1.24e+01)−
f_{19}	1.47e+00(1.88e−01)	1.36e+00(1.20e−01)+	1.61e+00(3.58e−01)−	1.84e+00(2.00e−01)−	1.44e+00(8.71e−02)≈	**1.27e+00(1.09e−01)**+
f_{20}	1.07e+01(9.74e−01)	1.05e+01(6.04e−01)+	1.06e+01(6.69e−01)≈	1.30e+01(6.33e−01)−	**1.04e+01(5.82e−01)**+	1.13e+01(4.14e−01)−
f_{21}	**3.00e+02(0.00e+00)**	3.09e+02(5.65e+01)−	3.02e+02(9.02e+01)−	3.05e+02(8.06e+01)−	3.04e+02(6.68e+01)−	2.94e+02(8.29e+01)≈
f_{22}	2.11e+02(8.47e+02)	**9.81e+01(2.52e+01)**+	1.17e+02(9.96e+00)+	3.09e+02(1.12e+02)−	9.39e+02(3.08e+01)−	1.03e+02(2.57e+01)+
f_{23}	3.61e+03(7.79e+02)	3.51e+03(4.11e+02)+	3.56e+03(6.12e+02)≈	6.74e+03(8.20e+02)−	**3.36e+03(4.01e+02)**+	4.36e+03(4.61e+02)−
f_{24}	2.09e+02(9.28e+00)	2.05e+02(5.29e+00)+	2.21e+02(9.28e+00)−	2.91e+02(7.08e+00)−	2.17e+02(1.57e+01)−	**2.04e+02(4.22e+00)**+
f_{25}	**2.54e+02(0.00e+00)**	2.59e+02(1.96e+01)−	2.57e+02(6.55e+00)−	2.99e+02(3.29e+00)−	2.74e+02(1.06e+01)−	2.55e+02(7.91e+00)≈
f_{26}	2.00e+02(4.41e+01)	2.02e+02(1.48e+01)−	2.18e+02(4.48e+01)−	3.56e+02(6.49e+01)−	2.15e+02(4.11e+01)−	**2.00e+02(3.06e−03)**+
f_{27}	9.15e+02(0.00e+00)	**3.88e+02(1.48e+01)**+	6.20e+02(1.01e+02)+	1.21e+03(7.42e+01)−	6.70e+02(2.40e+02)+	3.90e+02(9.12e+01)+
f_{28}	**3.00e+02(0.00e+00)**	**3.00e+02(0.00e+00)**≈	**3.00e+02(0.00e+00)**≈	**3.00e+02(0.00e+00)**≈	**3.00e+02(0.00e+00)**≈	**3.00e+02(0.00e+00)**≈
	+	13	7	1	9	10
	−	10	11	22	13	10
	≈	3	10	5	6	8

Table 6 Statistical ranking of *ADEwDC* and state-of-the-art DE algorithms while computing the minimum of the benchmark functions in CEC2013

	Unimodal	Multimodal	Composite functions
1	*SHADE*	**ADEwDC**	*dynNP-jDE*
2	**ADEwDC**	*JADE*	*SHADE*
3	*JADE*	*CoDE*	**ADEwDC**
4	*CoDE*	*dynNP-jDE*	*JADE*
5	*dynNP-jDE*	*SHADE*	*CoDE*
6	*EPSDE*	*EPSDE*	*EPSDE*

the best performance on unimodal problems. The good performance of *JADE* on the unimodal functions is consistent with previous results [28]. For the basic multimodal functions, *ADEwDC* performs relatively well, although for several of the problems including f_{13}, f_{18}, *JADE* and *CoDE* perform particularly well. For the complex, composite functions, the best performer is *dynNP-jDE* (possibly owing to its population size reduction strategy), followed by *SHADE*, *ADEwDC*, *JADE*, *CoDE* and *EPSDE*. Finally, based on the statistical data given in the bottom three rows of Table 5, *ADEwDC* achieves better performance than *CoDE*, *EPSDE* and *JADE*, and similar performance to *dynNP-jDE*, but does not surpass the performance of *SHADE* on these 28 problems. Still, *ADEwDC* is better suited to multimodal problems than *SHADE*. This is probably because the dynamic adjustment strategy gives *ADEwDC* extensive adaptability, which has advantages in dealing with multimodal problems.

4.4.2 Experiment on the CEC2005 benchmarks

To verify the validity and accuracy further, we compared *ADEwDC* with *SHADE* on the CEC2005 benchmarks [23]. CEC2005 provides many classic benchmark functions, which have been tested by many original authors of algorithms. Functions $f_1 \sim f_5$ are unimodal, while the others are multimodal. $f_6 \sim f_{12}$ are basic functions, $f_{13} \sim f_{14}$ are expanded functions, and $f_{15} \sim f_{25}$ are hybrid composition functions. For this experiment, comparative data were once again taken from [24].

In Table 7, it is shown that *ADEwDC* performs very well with multimodal problems, especially on the basic and hybrid composition functions, which coincides with the previous experimental analysis. Overall, *ADEwDC* is slightly better than SHADE. For the expanded functions, *ADEwDC* achieves similar performance to *SHADE*. Nonetheless, *ADEwDC* does not perform as well on the unimodal problems, especially for f_4, f_5, which is also consistent with the previous analysis. Generally, *ADEwDC* is ideal for multimodal optimization problems.

Table 7 Comparison of *ADEwDC* and *SHADE* while computing the minimum of benchmark functions in CEC2005. For all problems, we set the dimensionality $D = 30$, and the maximum number of objective function evaluations $D \times 10,000 = 300,000$. All results are averaged over 25 trials

F	ADEwDC Mean(Std)	SHADE Mean(Std)
f_1	**0.00e+00(0.00e+00)**	**8.05e−18(1.27e−17)**≈
f_2	**0.00e+00(2.66e−05)**	**2.53e−17(9.74e−18)**≈
f_3	**6.31e−03(5.38e−03)**	1.18e+04(8.43e+03)−
f_4	3.90e+02(3.09e+02)	**8.45e−10(7.03e−09)**+
f_5	2.35e+03(1.62e+03)	**1.18e−03(8.87e−03)**+
f_6	**0.00e+00(1.99e+00)**	2.79e−01(1.02e+00)−
f_7	**1.72e−02(2.53e−02)**	**9.88e−03(6.98e−03)**≈
f_8	**2.00e+01(0.00e+00)**	2.03e+01(3.69e−01)−
f_9	2.81e+01(5.47e+00)	**0.00e+00(0.00e+00)**+
f_{10}	5.63e+01(1.06e+01)	**2.36e+01(3.47e+00)**+
f_{11}	**2.26e+01(3.06e+00)**	2.66e+01(1.93e+00)−
f_{12}	**4.18e+02(3.00e+02)**	1.60e+03(2.34e+03)−
f_{13}	3.51e+00(1.10e+00)	**1.36e+00(9.64e−02)**+
f_{14}	**1.21e+01(2.40e−01)**	1.24e+01(2.55e−01)−
f_{15}	**3.05e+02(2.35e+02)**	3.58e+02(9.34e+01)−
f_{16}	2.91e+02(1.87e+02)	**7.40e+01(8.82e+01)**+
f_{17}	1.96e+02(1.33e+01)	**9.20e+01(6.89e+01)**+
f_{18}	**8.46e+02(1.56e+01)**	9.02e+02(1.82e+01)−
f_{19}	**8.44e+02(1.88e+00)**	9.05e+02(1.07e+01)−
f_{20}	**8.39e+02(1.22e+00)**	9.04e+02(1.07e+01)−
f_{21}	**5.00e+02(0.00e+00)**	5.03e+02(3.00e+01)−
f_{22}	**7.90e+02(1.58e+02)**	8.78e+02(1.33e+01)−
f_{23}	5.53e+02(1.63e+01)	**5.38e+02(4.03e+01)**+
f_{24}	2.41e+02(2.73e+01)	**2.00e+02(0.00e+00)**+
f_{25}	**2.10e+02(3.76e−01)**	**2.09e+02(1.46e−01)**≈
	+	9
	−	12
	≈	4

4.4.3 High dimensional experiment on the CEC2013 benchmarks

We also executed *ADEwDC* on the CEC2013 benchmarks, setting the number of dimensions $D = 50$ as a high-dimensional case to prove the universality and robustness of the proposed algorithm. The maximum number of objective function calls per run was set as $D \times 10,000$ (that is, 500,000). From Table 8, it can clearly be seen that *ADEwDC* is also suitable for high-dimensional problems, especially for multimodal optimization functions, which is consistent with our previous experimental analysis and conclusions.

Table 8 *ADEwDC* applied to the CEC2013 benchmarks, with dimensionality $D = 50$, and maximum number of objective function evaluations $D \times 10,000 = 500,000$. The results of this high-dimensional test confirm the universality and robustness of ADEwDC, with all results averaged over 51 trials

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Best	0.00e+00	6.64e−02	1.60e+07	3.76e+04	0.00e+00	0.00e+00	4.93e+01	2.09e+01	3.19e+01	0.00e+00
Mean	0.00e+00	3.19e−01	5.14e+07	4.40e+04	0.00e+00	0.00e+00	6.06e+01	2.11e+01	3.88e+01	3.32e−02
Std	0.00e+00	3.14e−01	7.10e+07	5.00e+03	0.00e+00	0.00e+00	5.99e+00	5.56e−02	4.68e+00	2.35e−02
	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
Best	6.66e+01	9.65e+01	2.58e+02	1.39e+03	5.63e+03	2.45e−01	1.11e+02	1.24e+02	1.08e+01	1.79e+01
Mean	9.65e+01	1.13e+02	3.11e+02	2.68e+03	6.40e+03	4.90e−01	1.29e+02	1.46e+02	1.72e+01	2.04e+01
Std	2.19e+01	1.64e+01	3.29e+01	8.81e+02	7.97e+02	2.54e−01	1.99e+01	1.63e+01	5.25e+00	1.50e+00
	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}		
Best	8.36e+02	1.22e+03	6.74e+03	2.42e+02	2.50e+02	2.00e+02	7.88e+02	4.00e+02		
Mean	8.36e+02	2.46e+03	9.36e+03	2.56e+02	2.70e+02	2.00e+02	8.72e+02	4.00e+02		
Std	1.64e+02	9.76e+02	2.06e+03	2.69e+01	1.02e+01	5.71e+01	2.21e+02	0.00e+00		

5 Conclusions and future work

A novel adaptive DE algorithm with directional strategy and cloud model was presented in this paper. The DE algorithm is known to be an efficient and powerful optimization algorithm, which is used widely in scientific and engineering fields. Considering that most DE algorithms still suffer from a number of problems such as difficult parameter setting, slow convergence rate, and premature convergence, the proposed algorithm utilizes a *dynamic adjustment strategy* to select evolutionary variants and set control parameters based on the constructed topology, adopts a *directional strategy* to evolve outstanding individuals and improve the convergence speed of the algorithm, and maintains the diversity of the population by employing the *cloud model*. Results of the experiments confirm that ADEwDC achieves good performance in terms of convergence, stability, and precision, which greatly contributes to overcoming the low efficiency in conventional DE algorithms. Meanwhile, *ADEwDC* effectively avoids falling into a local optimum. Our future work involves strengthening the theoretical analysis of *ADEwDC*, further improving its accuracy and stability, and generalizing it to solve constraint and multi-objective optimization problems in practical applications.

Acknowledgments This work was supported by the National Natural Science Foundation of China (No. 61103170, 51305142, 61305085), the Program for Prominent Young Talent in Fujian Province University (No. JA12005), and the Promotion Program for Young and Middle-aged Teachers in Science and Technology Research at Huaqiao University (No. ZQN-PY211).

References

1. Brest J, Mauec MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247
2. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657
3. Cai YQ, Wang JH (2013) Differential evolutionwith neighborhood and direction information for numerical optimization. IEEE Trans Cybern 43(6):2202–2215
4. Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. Evol Comput 15(1):4–31
5. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood based mutation operator. IEEE Trans Evol Comput 13(3):526–553
6. Ding JL, Liu J, Chowdhury KR, Zhang WS, Hu QP, Lei J (2014) A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization. Neurocomputing 137:261–267
7. Dorronsoro B, Bouvry P (2011) Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Trans Evol Comput 15(1):67–98
8. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141
9. Gao XZ, Wang XL, Ovaska SJ (2009) Fusion of clonal selection algorithm and differential evolution method in training cascade-correlation neural network. Neurocomputing 72(10–12):2483–2490
10. Gao Y (2009) An optimization algorithm based on cloud model. In: 2009 international conference on computational intelligence and security, pp 84–87
11. Li DY, Du Y (2005) Artificial intelligence with uncertainty (in Chinese). National Defense Industry Press, Beijing
12. Li DY, Liu CY (2005) Study on the universality of the normal cloud model. Eng Sci 3(2):18–24

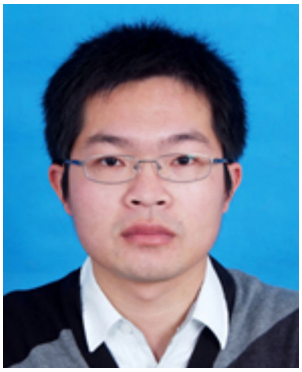
13. Li DY, Meng HJ, Shi XM (1995) Membership clouds and membership cloud generators (in chinese). J Comput Res Dev 32(6):15–20
14. Li DY, Liu CY, Gan WY (2009) A new cognitive model: Cloud model. Int J Intell Syst 24(3):357–375
15. Liang JJ, Qu BY, Suganthan PN, Hernández-Díaz AG (2013) Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Tech. rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China
16. Liu G, Li YX, Nie X, Zheng H (2012) A novel clustering-based differential evolution with 2 multi-parent crossovers for global optimization. Appl Soft Comput 12(2):663–681
17. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696
18. Omran MGH, Salman A, Engelbrecht AP (2005) Self-adaptive differential evolution. Proceedings part I international conference, CIS 2005. Springer Berlin Heidelberg, pp 192–199
19. Pan QK, Suganthan PN, Ling W, Liang G, Mallipeddi R (2011) A differential evolution algorithm with self-adapting strategy and control parameters. Comput Oper Res 38(1):394–408
20. Price KV (1999) An introduction to differential evolution. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, Ltd., UK, pp 79–108
21. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
22. Storn R, Price K (1997) Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. J Glob Optim 11(4):341–359
23. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University
24. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE congress on evolutionary computation (CEC)
25. Vafashoar R, Meybodi MR, Azandaryani AHM (2012) Cla-de: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell 36(3):735–748
26. Varadarajan M, Swarup KS (2008) Differential evolutionary algorithm for optimal reactive power dispatch. Electr Power Energy Systs 30(8):435–441
27. Wang X, Zhao SG (2013) Differential evolution algorithm with self-adaptive population resizing mechanism. Math Probl Eng 2013(2013):1–14
28. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
29. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bulletin 1(6):80–83
30. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
31. Yildiz AR (2013) Hybrid taguchi-differential evolution algorithm for optimization of multi-pass turning operations. Appl Soft Comput 13(3):1433–1439
32. Zaharie D (2003) Control of population diversity and adaptation in differential evolution algorithms. In: Matousek R, Osmera P (eds) 2003 9th international conference on soft computing, pp 41–46
33. Zhan ZH, Zhang J, Li Y, Shi YH (2011) Orthogonal learning particle swarm optimization. IEEE Trans Evol Comput 15(6):832–847
34. Zhang JQ, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
35. Zhang JZ, Ding XM (2011) A multi-swarm self-adaptive and cooperative particle swarm optimization. Eng Appl Artif Intell 24(6):958–967
36. Zhao SZ, Suganthan PN, Das S (2011) Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput 15(11):2175–2185
37. Zhao ZQ, Gou J, Wang J (2010) Directional evolutionary algorithm based on fitness gradient of individuals (in chinese). Pattern Recognit Artif Intell 23(1):29–37
38. Zheng YJ, Ling HF, Xue JY (2014) Ecogeography-based optimization: Enhancing biogeography-based optimization with ecogeographic barriers and differentiations. Comput Oper Res 50:115–127
39. Zhu CM, Ni J (2012). Cloud model-based differential evolution algorithm for optimization problems. In: 2012 6th international conference on internet computing for science and engineering, pp 55–59

Jin Gou received the B.E. degree and M.E. degree in computer science and technology from Fuzhou University, China in 1999 and 2002. He received the Ph.D. degree in Computer Science and Technology from Zhejiang University, China in 2006. Currently, he is an Associate Professor at Huaqiao University, China. His research interests include knowledge fusion and artificial intelligence.

Wang-Ping Guo received the graduate degree in computer science and technology from Huaqiao University, China in 2012. Currently, he is a graduate student at Huaqiao University, China. His research interests include artificial intelligence and evolutionary computation.

Feng Hou received the graduate degree in computer science and technology from Huaqiao University, China in 2012. Currently, he is a graduate student at Huaqiao University, China. His research interests include artificial intelligence and data mining.

Cheng Wang received the Ph.D. degree in mechanics from Xi'an jiaotong University, China in 2012. Currently, he is a lecturer at Huaqiao University, China. His research interests include signal processing and data mining.



Yi-Qiao Cai received his Ph.D. degree from Sun Yat-sen University, Guangzhou, China, in 2012. In 2012, he joined Huaqiao University, Xiamen, China, where he is currently a Lecturer with the College of Computer Science and Technology. He is now interested in differential evolution, multi-objective optimisation, and other evolutionary computation techniques.