Applied Soft Computing Journal 81 (2019) 105496

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Self-adaptive mutation differential evolution algorithm based on particle swarm optimization

Shihao Wang [a,*], Yuzhen Li [a], Hongyu Yang [b]

[a] Department of Information Security, Henan Police College, Zhengzhou 450046, China
[b] National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu 610065, China

HIGHLIGHTS

- A self-adaptive mutation DE based on particle swarm optimization is proposed.
- An elite archive strategy is introduced to improve DE/rand/1 mutation strategy.
- A self-adaptive mutation strategy is presented in DEPSO.
- The effect of the parameters on DEPSO is experimentally investigated.
- The DEPSO is applied to arrival flights scheduling.

ARTICLE INFO

Article history:
Received 14 September 2018
Received in revised form 6 May 2019
Accepted 10 May 2019
Available online 14 May 2019

Keywords:
Differential evolution
Particle swarm optimization
Self-adaptive mutation
Elite archive
Arrival flights scheduling

ABSTRACT

Differential evolution (DE) is an effective evolutionary algorithm for global optimization, and widely applied to solve different optimization problems. However, the convergence speed of DE will be slower in the later stage of the evolution and it is more likely to get stuck at a local optimum. Moreover, the performance of DE is sensitive to its mutation strategies and control parameters. Therefore, a self-adaptive mutation differential evolution algorithm based on particle swarm optimization (DEPSO) is proposed to improve the optimization performance of DE. DEPSO can effectively utilize an improved DE/rand/1 mutation strategy with stronger global exploration ability and PSO mutation strategy with higher convergence ability. As a result, the population diversity can be maintained well in the early stage of the evolution, and the faster convergence speed can be obtained in the later stage of the evolution. The performance of the proposed DEPSO is evaluated on 30-dimensional and 100-dimensional functions. The experimental results indicate that DEPSO can significantly improve the global convergence performance of the conventional DE and thus avoid premature convergence, and its average performance is better than those of the conventional DE, PSO and the compared algorithms. Moreover, DEPSO is applied to solve arrival flights scheduling and the optimization results show that it can optimize the sequence and decrease the delay time.

1. Introduction

Differential evolution (DE) algorithm, first proposed by Stron and Price [1,2], is a population based stochastic parallel search method. Due to its simplicity and effectiveness, DE has been successfully applied in diverse fields of science and engineering such as pattern recognition [3–5], image processing [6–8], power systems [9–12], industrial control [13,14] and functions optimization [15,16].

However, the performance of DE depends on its mutation strategy and the associated parameter settings (population size NP, scaling factor F, and crossover rate CR). In general, the appropriate mutation strategy and control parameter values should be determined to obtain the best optimization result before using DE to solve a specific optimization problem. However, choosing appropriate mutation strategy and its associated parameter values generally requires time-consuming trial-and-error, especially when solving high-dimensional and unknown optimization problems [17]. The choices for the fixed mutation strategy and parameter settings are simple, but only suitable for some specific optimization problems. Moreover, even for a specific problem, the best mutation strategy and control parameter settings may vary during different stages of the evolution process [18].

To solve the above problems, many improved DE variants were proposed by adaptively choosing mutation strategy and tuning parameter values, such as FADE (fuzzy adaptive DE) [19], jDE

* Corresponding author.
E-mail address: bhwang001@126.com (S. Wang).

https://doi.org/10.1016/j.asoc.2019.105496
1568-4946/

(self-adapted control parameters) [20], aDE (adaptive DE) [21], SaDE (self-adaptive DE) [22], JADE (DE/current-to-pbest mutation strategy with optional external archive and adaptive control parameters) [23], CoDE (composite DE) [24], EPSDE (ensemble of mutation strategies and control parameters with DE) [25], DMPSADE (self-adaptive DE algorithm with discrete mutation parameters) [26], MPEDE (multi-population ensemble DE) [10], IMSaDE (self-adaptive DE with improved mutation strategy) [27]. However, the adjustment timing of adaptive mutation strategy and control parameter strategy on most algorithms has the randomness or depends too much on the history experience, and is lacks specific guidance.

Particle swarm optimization (PSO), introduced by Kennedy and Eberhart [28], is a parallel search technique for global optimization problems. It is simple and easy to implement. The search strategy based on individual best solution and global best solution found in history makes PSO converge faster in the early stage of the evolution, but the rapidly reduced population diversity is more likely to lead to premature convergence in the later stage of the evolution.

To overcome the drawbacks in PSO and DE, the hybridization of the two algorithms has attracted persistent attention, and many hybrid methods are proposed, such as PSO–DE [29], HCPSODE (hybrid chaotic PSO with DE) [30], DEMPSO (DE and modified PSO) [31], and EPSODE (ensemble PSO and DE) [32]. Although these hybrid algorithms can improve the optimization performance of the conventional DE and PSO, more computational resources needed and premature convergence are still the major problems.

To tackle these problems and balance global exploration and local exploitation of DE, a self-adaptive mutation differential evolution algorithm based on particle swarm optimization (DEPSO) is proposed in this paper. In the early stage of the evolution, DEPSO takes full advantage of DE/rand/1 mutation strategy improved using an elite archive strategy to maintain the population diversity, and thus the individuals are able to explore a relatively large region and avoid premature convergence. In the later stage of the evolution, DEPSO can effectively utilize PSO mutation strategy with faster convergence ability to enhance the convergence speed and precision. Additionally, the parameters adaptation strategy based on individual evolution status can track the evolution process of each individual in real-time, and thus the parameter settings are more reasonable. The cooperation between self-adaptive mutation strategy and parameter adaptation strategy ensures that DEPSO has higher convergence performance and good robustness.

Our work is different from previous studies in three aspects. First, all previous work adopts the original DE/rand/1 mutation strategy, while in DEPSO DE/rand/1 is improved using an elite archive strategy to enhance its convergence speed and precision. Second, in previous work, most hybrid structures based on PSO and DE are sequential or parallel, which may require more computational resources. However, in our study, DEPSO retains the original operation of DE and only adds PSO mutation strategy to its mutation operation. DE/rand/1 and PSO mutation strategies for each individual are mutually exclusive and can be adaptively selected as the algorithm proceeds. Therefore, no additional resources are needed, and the operation is simple and easy to implement. Third, the selection of mutation strategies for each individual in previous literature has strong randomness during the whole evolution process, thus lacking promising guidance. By contrast, the proposed DEPSO can make full use of the improved DE/rand/1 mutation strategy in the early stage to explore in all directions and expand the search regions. In the later stage, PSO mutation strategy is used to carefully search the promising regions found in the early stage to improve the local exploitation ability.

30-dimensional and 100-dimensional test functions were used to investigate the performance of the proposed DEPSO. The experimental results show that the average performance of DEPSO is better than the conventional DE, PSO, jDE [20], aDE [21], SaDE [22], JADE [23], CoDE [24], EPSDE [25], DEMPSO [31], IMSaDE [27] and a canonical PSO [33]. Moreover, arrival flights scheduling problem was used to evaluate the practical optimization ability of DEPSO. The optimization results show that DEPSO has stronger global search ability and can effectively decrease the delay time of arrival flights.

The rest of the paper is organized as follows: Section 2 briefly introduces the conventional DE and PSO. Section 3 reviews the related work on DE in detail. Section 4 presents the proposed DEPSO. Experimental results are reported in Section 5. Section 6 presents the application of DEPSO to arrival flights scheduling. Finally, Section 7 concludes the paper.

2. Description of algorithms

2.1. Differential evolution

DE is a population-based stochastic search method which works through three operations, including mutation, crossover and selection. DE utilizes NP D-dimensional parameter vectors $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \mathbf{X}_{2,G}, \ldots, \mathbf{X}_{NP,G}\}$ where $\mathbf{X}_{i,G} = \{x^1_{i,G}, x^2_{i,G}, \ldots, x^D_{i,G}\}$, $i = 1, 2, \ldots, NP$ denotes the ith solution (called a individual) in the Gth generation. Suppose that the optimization problem to be minimized is $\min_{\mathbf{x}_i \in \Re^D} f(\mathbf{x}_i)$, and the feasible solution space is $\Omega = \prod_{j=1}^{D}[X^j_{\min}, X^j_{\max}]$. The procedure of DE is as follows.

(1) Population initialization. The population size NP, the dimension of the problem D, scaling factor F, crossover rate CR and maximum number of evolution generations $G_{\max}$ are determined beforehand, and then set $G = 0$. The initial population $\mathbf{P}_0 = \{\mathbf{X}_{1,0}, \mathbf{X}_{2,0}, \ldots, \mathbf{X}_{NP,0}\}$ is randomly generated from the prescribed solution space.

(2) Mutation. During the evolution, DE employs the mutation operation to generate a mutation vector (individual) $\mathbf{V}_{i,G+1} = \{v^1_{i,G+1}, v^2_{i,G+1}, \ldots, v^D_{i,G+1}\}$ with respect to each target individual $\mathbf{X}_{i,G}$ in the parent population. The five frequently used mutation strategies are shown as follows.

(1) DE/rand/1

$$\mathbf{V}_{i,G+1} = \mathbf{X}_{r1,G} + F \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) \tag{1}$$

(2) DE/rand/2

$$\mathbf{V}_{i,G+1} = \mathbf{X}_{r1,G} + F \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) + F \cdot (\mathbf{X}_{r4,G} - \mathbf{X}_{r5,G}) \tag{2}$$

(3) DE/best/1

$$\mathbf{V}_{i,G+1} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) \tag{3}$$

(4) DE/best/2

$$\mathbf{V}_{i,G+1} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) + F \cdot (\mathbf{X}_{r3,G} - \mathbf{X}_{r4,G}) \tag{4}$$

(5) DE/current-to-best/1

$$\mathbf{V}_{i,G+1} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) \tag{5}$$

where the indices $r1$, $r2$, $r3$, $r4$ and $r5$ are mutually exclusive integers randomly chosen within the range $[1, NP]$, and are also different from the index i ($r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$). The scaling factor F is a positive constant, which amplifies the difference vector. $\mathbf{X}_{best,G}$ is the best individual with the minimum fitness (objective function value) in the population at generation G.

(3) Crossover. After mutation, with respect to the target individual $\mathbf{X}_{i,G}$ and its corresponding mutation vector, DE employs a

binomial crossover to produce a trail vector (individual) $\mathbf{U}_{i,G+1} = \left\{u^1_{i,G+1}, u^2_{i,G+1}, \ldots, u^D_{i,G+1}\right\}$.

$$u^j_{i,G+1} = \begin{cases} v^j_{i,G+1} & \text{if } r_j \le CR \text{ or } j{=}j_{rand} \\ x^j_{i,G} & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, D \tag{6}$$

where the crossover rate $CR \in [0, 1]$ is a prescribed constant. r_j is a uniform random number within the range $[0, 1]$. j_{rand} is a randomly chosen integer in $[1, D]$, which ensures that the trail vector $\mathbf{U}_{i,G+1}$ can get at least one element from the mutation vector $\mathbf{V}_{i,G+1}$, and thus avoid evolutionary stagnation.

However, some elements of the trail vector $\mathbf{U}_{i,G+1}$ may deviate from the feasible solution space due to the impact of mutation operation. Therefore, the infeasible elements should be reset as follows.

$$u^j_{i,G+1} = \begin{cases} X^j_{min} + rand(0,1)\cdot(X^j_{max} - X^j_{min}) & \text{if } u^j_{i,G+1} \notin [X^j_{min}, X^j_{max}] \\ u^j_{i,G+1} & \text{otherwise} \end{cases} \tag{7}$$

(4) Selection. DE employs a greedy selection scheme to evaluate the trail vector $\mathbf{U}_{i,G+1}$. If the fitness of the trail vector $\mathbf{U}_{i,G+1}$ is less than or equal to the corresponding target individual $\mathbf{X}_{i,G}$, $\mathbf{U}_{i,G+1}$ will replace $\mathbf{X}_{i,G}$ and enter the next generation.

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G+1} & \text{if } f(\mathbf{U}_{i,G+1}) \le f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G} & \text{otherwise} \end{cases} \tag{8}$$

Set $G = G{+}1$ when finishing the above steps (2)∼(4), and then these steps are repeated until a termination criterion ($G = G_{\max}$) is satisfied.

2.2. Particle swarm optimization

In PSO, a scheme based on velocity-position is used to perform the search process, and each particle (individual) can be treated as a point in a D-dimensional space. The position and velocity of the ith particle at generation G are represented as $\mathbf{X}_{i,G} = \left\{x^1_{i,G}, x^2_{i,G}, \ldots, x^D_{i,G}\right\}$ and $\mathbf{S}_{i,G} = \left\{s^1_{i,G}, s^2_{i,G}, \ldots, s^D_{i,G}\right\}$, respectively. During the evolution, each particle tracks individual best solution $\mathbf{X}_{pbest}$ and global best solution $\mathbf{X}_{gbest}$ to update its own velocity $\mathbf{S}_{i,G}$ and position $\mathbf{X}_{i,G}$.

$$\mathbf{S}_{i,G+1} = \mathbf{S}_{i,G} + c_1 \cdot r_1 \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{i,G}) + c_2 \cdot r_2 \cdot (\mathbf{X}_{gbest,G} - \mathbf{X}_{i,G}) \tag{9}$$

$$\mathbf{X}_{i,G+1} = \mathbf{X}_{i,G} + \mathbf{S}_{i,G+1} \tag{10}$$

where c_1 and c_2 are the learning factors, and generally set to $c_1 = c_2 = 2$. r_1 and r_2 are two random numbers chosen from the range $[0, 1]$. To obtain a tradeoff between local search ability and global search ability, Shi and Eberhart [34] brought an inertia weight ω into Eq. (9) as shown in Eq. (11).

$$\mathbf{S}_{i,G+1} = \omega \cdot \mathbf{S}_{i,G} + c_1 \cdot r_1 \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{i,G}) + c_2 \cdot r_2 \cdot (\mathbf{X}_{gbest,G} - \mathbf{X}_{i,G}) \tag{11}$$

$$\omega = \omega_{max} - \frac{G}{G_{max}}(\omega_{max} - \omega_{min}) \tag{12}$$

where G is the current number of evolution generations and $G_{\max}$ is the maximum number of evolution generations. $\omega_{\max}$ and $\omega_{\min}$ are the maximum and the minimum for inertia weight ω, respectively. In general, they are set to $\omega_{\max} = 0.9$ and $\omega_{\min} = 0.4$.

3. Related work

As an important branch of evolutionary algorithm (EA), DE has attracted much attention due to its simplicity and efficiency. However, the performance of DE depends on its mutation strategy and control parameters. Furthermore, like other EAs, DE also has premature convergence. Since DE was proposed, researchers have proposed many enhanced DE variants to improve the performance of the conventional DE.

3.1. Population initialization

Rahnamayan, Tizhoosh and Salama [35] proposed an opposition
-based DE (ODE) algorithm that employs opposition-based learning for population initialization. Ali, Pant and Abraham [36] introduced a non-linear simplex DE (NSDE) algorithm wherein non-linear simplex method in conjugation of pseudo random number is used to generate the initial population. Gong, Cai and Jiang [37] utilized the orthogonal design method to generate the initial population and design the crossover operator for DE. The experimental results show that their method can find the optimal or close-to-optimal solutions. In [38], a chaotically initialized differential evolution (CIDE) algorithm was proposed. CIDE uses chaotic maps to initialize the population and seven chaotic maps are analyzed in the benchmark problems. Ali, Pant and Abraham [39] presented two schemes for initializing the population of DE, which are based on quadratic interpolation (QI) and non-linear simplex method (NSM), respectively. The numerical results indicate that the proposed schemes can improve the performance of DE in terms of convergence rate and average CPU time. Poikolainen, Neri, and Caraffini [40] introduced a pre-processing procedure (cluster-based population initialization) for DE which consists of three consecutive stages. The results indicate that the pre-processing method is an efficient software component which can consistently improve the performance of DE.

3.2. Mutation strategy

DE has five commonly used mutation strategies (i.e., DE/rand/1, DE/best/1, DE/rand/2, DE/best/2 and DE/current-to-best/1). Generally, different mutation strategies are chosen to solve different optimization problems according to the nature of the problem. Qin, Huang and Suganthan [18] proposed a self-adaptive DE (SaDE), in which four mutation strategies (i.e., DE/rand/1, DE/rand-to-best/2, DE/rand/2 and DE/current-to-rand/1) are incorporated into a strategy candidate pool. In SaDE, with respect to each target individual in the parent population, one mutation strategy is chosen from the strategy candidate pool according to the probability learned from its success rate in generating better offspring. Furthermore, SaDE utilizes two normal distributions $N(0.5, 0.3)$ and $N(CR_m, 0.1)$ to generate F and CR, respectively. CR_m is adapted according to the successful crossover rates in previous generations. Wang, Cai and Zhang [24] combined three mutation strategies (i.e., DE/rand/1, DE/rand/2 and DE/current-to-rand/1) with three parameter settings to propose a composite DE (CoDE) algorithm. In CoDE, with respect to each target vector, three trail vectors are generated and the best one will enter the next generation if it is better than the corresponding target vector. In EPSDE [25], three mutation strategies (i.e., DE/best/2, DE/rand/1 and DE/current-to-rand/1) are incorporated into a strategy pool, and a set of parameter values (i.e., $F \in [0.4, 0.9]$ in steps of 0.1 and $CR \in [0.1, 0.9]$ in steps of 0.1) are incorporated into a pool of parameter settings. However, the control parameters in CoDE and EPSDE are fixed and not adapted. DMPSADE proposed by Fan and Yan [26] employs five

mutation strategies (i.e., DE/rand/1, DE/rand-to-best/2, DE/rand-to-best/1, DE/best/2 and DE/rand/2) to constitute a candidate pool of mutation strategy. Then a roulette wheel scheme is used to determine the mutation strategy for each target individual according to the cumulative probability of each mutation strategy. In addition, each element of each individual has its own scaling factor F, and each individual has its own crossover rate CR. These control parameters are automatically adjusted according to normal distribution. Wu et al. [10] proposed a multi-population ensemble DE (MPEDE) which consists of three mutation strategies (i.e., DE/rand/1, DE/current-to-rand/1 and DE/current-to-$p$best/1). The population is partitioned into multiple indicator subpopulations and each indicator subpopulation is assigned to a mutation strategy. During the evolution, the better mutation strategies can obtain more computational resources and the parameter values of each strategy are adapted independently. IMSaDE proposed by Wang et al. [27] adopts an elite archive strategy to improve DE/rand/2 mutation strategy, in which some individuals involving mutation operation are from elite population and the remainders are from non-elite population, but the mutation strategy is fixed and not adapted.

3.3. Control parameters adaptation

Liu and Lampinen [19] introduced a fuzzy adaptive DE (FADE) that employs fuzzy logic controllers to adaptively generate new parameters values, but these values are not applied at individual level and the mutation strategy is not adapted. Brest et al. [20] proposed a jDE algorithm with self-adapted parameters wherein the scaling factor F and crossover rate CR are applied at individual level. In jDE, the parameters values lead to better individuals that are more likely to survive and enter the next generation. Otherwise, they will be randomly adjusted according to two constants $\tau_1 = 0.1$ and $\tau_2 = 0.1$. Nasimul, Danushka and Hitoshi [21] proposed an adaptive DE (aDE) that is similar to jDE. The difference is that the updating of the parameter values depends on whether the offspring is superior to the average individual of the parent population. However, neither jDE nor aDE adopts the adaptive mutation strategy. In view of the fast but less reliable convergence performance of DE/current-to-best/1, Zhang and Sanderson [23] proposed a new mutation strategy DE/current-to-$p$best/1 with optional archive (JADE), which utilizes the recently explored inferior solutions to improve the population diversity. For each individual in the population, Cauchy distribution and normal distribution are used to generate F and CR, respectively. Asafuddoula, Ray, and Sarker [41] introduced an adaptive hybrid DE algorithm (AH-DEa), in which binomial crossover is used to explore in the early stage while exponential crossover is used to exploit in the later stage. In AH-DEa, the crossover rate can be adaptively adjusted based on the success of trail individuals. Furthermore, some improved DE algorithms also focus on the population size NP. Teo [42] presented a first attempt at self-adaptively tuning the population size. The experimental results indicate that DE with self-adapted population can obtain more competitive results than the conventional DE. Brest and Maucec [43] proposed a method for gradually reducing population size as the algorithm proceeds. Zhu et al. [44] introduced an adaptive population tuning scheme (APTS) for DE to adjust the population size according to the solution-searching status. Piotrowski [45] reviewed the opinions regarding population size for DE and recommended the use of adaptive population size, especially when solving higher-dimensional and real-world problems.

3.4. Hybrid DE algorithms

Hybridization is one of the most efficient strategies to improve the performance of optimization algorithms [46]. Sun, Zhang and Tsang [47] employed DE and estimation of distribution algorithm (EDA) to propose a DE/EDA algorithm, in which global information obtained by EDA and differential information obtained by DE are used to produce promising solutions. The simulation results show that DE/EDA is better than DE and EDA. Wang, Xu and Li [48] proposed a hybrid NMDE algorithm by combining DE and Nelder–Mead (NM) simplex search. In NMDE, evolutionary search based on DE and local search based on NM simplex are fused and satisfactory performance is achieved. The numerical simulation demonstrates that NMDE is effective and robust. Ponsich and Coello [49] introduced a DE-TS algorithm by hybridizing DE with tabu search (TS) to solve the job-shop scheduling problem. The computational experiments show that the proposed algorithm is competitive. Guo et al. [50] presented an enhanced self-adaptive DE (ESADE), in which simulated annealing is involved in the selection operation of DE to improve global search ability. The computational results show that ESADE performs better than the other compared algorithms.

In addition, there are many hybrid DE algorithms that combine with PSO. In [29], a hybrid PSO–DE algorithm was presented. PSO–DE performs first PSO evolution, then DE/rand/1, DE/current-to-best/1 and DE/rand/2 mutation strategies are used to produce three offspring for the best position of each particle respectively. Also, the personal best of the particle is compared with three offspring using a selection criterion. The hybrid method speeds up the convergence and improves the performance, but it needs more computational resources and the parameter settings for DE are fixed and not adapted during the run of the algorithm. Pant, Thangaraj and Abraham [51] proposed a novel hybrid version of DE and PSO (namely DEPSO-PTA), which uses the conventional DE to generate the trial vector. If the trial vector is better than the corresponding target vector, then the trial vector will be included in the next population. Otherwise the algorithm will activate PSO to produce a new candidate solution. Although the algorithm retains the advantages of both DE and PSO, it may be time-consuming and the parameter settings in DE are still fixed and thus decrease the applicability of the algorithm. Differing from PSO–DE and DEPSO-PTA, Sayah and Hamouda [52] incorporated PSO into the mutation operator of DE, and thus two mutant vectors and two trial vectors are generated for each target vector. Then the best vector with the best fitness will be selected. Obviously, this hybrid algorithm increases the time and space complexity of the algorithm to a certain extent and the optimal control parameter values are obtained by the tuning process. In [30], a hybrid chaotic PSO with DE (HCPSODE) was proposed. HCPSODE uses an iterative chaotic map with infinite collapses to replace the random numbers in the velocity of particles. The algorithm takes PSO as the main structure. When the population settles into stagnation state, DE/rand/1/bin will be performed to escape from the local optimum. However, a longer running process and fixed parameter settings still exist. Tian, Ren and Zhou [53] incorporated mutation and crossover operators of DE into PSO and presented an improved mutation operator instead of the particle's velocity and position of PSO. Also, crossover operation is performed by randomly choosing neighboring individual. In this hybrid algorithm, the selection of control parameters is specified for the entire run. In [31], a method combined by DE and Modified PSO (namely DEMPSO) was presented. The procedure of DEMPSO is similar to that of DEPSO-PTA [51], and the difference is that PSO is performed only when the fitness of global best position is less than the specified judge parameter. Wang et al. [32] presented a new ensemble PSO and DE method (namely EPSODE)

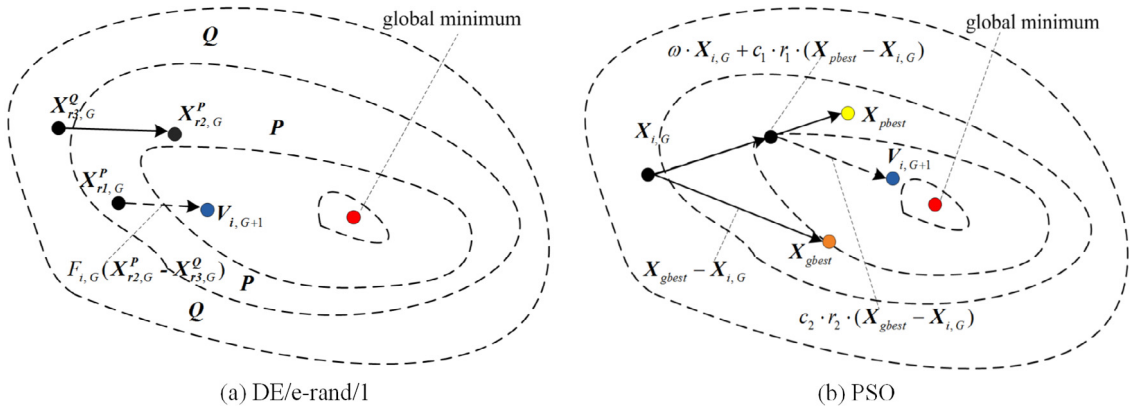

Fig. 1. Illustration of mutation strategy adopted in DEPSO.

to improve the search ability. In EPSODE, two subpopulations are generated by PSO and DE at the beginning of each generation, and then they are compared with the initial population or last population and the better individuals are retained to constitute new population that is used to perform the procedure of DE. The key point of this method is that two subpopulations are generated before each iteration, which will increase the space complexity of the algorithm.

4. DEPSO Algorithm

Generally speaking, the most suitable mutation strategy should be selected in advance to obtain satisfactory optimization performance when using DE to solve a specific optimization problem. However, for different optimization problems, choosing suitable mutation strategy requires the time-consuming trail-and-error. Mutation strategies (i.e., DE/best/1, DE/best/2 and DE/current-to-best/1) relying on the best solution usually have stronger local exploitation ability and faster convergence speed, but are more likely to suffer from premature convergence when solving multi-modal problems [22,23]. DE/rand/1 and DE/rand/2 mutation strategies relying on the random individual usually have stronger global exploration ability, while their convergence speed is slower [22]. Therefore, they are more suitable for multi-modal problems.

Global exploration aims to obtain the promising information regarding objective function from the entire search space during the evolution process, and thus it is able to locate the regions where the local/global optimal solution may be located. Local exploitation aims to carefully search the promising region where the local/global best solution is located, and thus it is able to find the local optimal solution and even the global optimal solution.

For population-based evolutionary algorithms, the coordination between population diversity and convergence speed is an external representation of a tradeoff between global exploration and local exploitation. If the tradeoff strategy is excessively inclined to global exploration, the algorithm may tend to the region where the global optimal solution is located. However, it often requires more computational resources, and the convergence speed will be slower. Conversely, the fact that the tradeoff strategy is excessively inclined to local exploitation may lead to premature convergence.

Most of the intelligent optimization algorithms adopt the transformation from global exploration to local exploitation. However, the global exploration ability will decrease as the iteration increases. If the global optimum has not yet been found and the exploration ability has been lost, the algorithm will fall into the local optimum and cannot achieve global optimization. Therefore, a tradeoff strategy is needed to achieve the balance between global exploration and local exploitation.

Based on the above analysis, this paper presents a self-adaptive mutation DE algorithm based on PSO (DEPSO). In the early stage of the evolution, DEPSO has good population diversity which is beneficial to expand search region and obtain more useful information. In the later stage of the evolution, DEPSO has faster convergence speed and can make full use of the information accumulated in the early stage to improve the precision of optimization results.

4.1. Self-adaptive mutation strategy in DEPSO

Refer to Eq. (1) (DE/rand/1) and (2) (DE/rand/2), the right sides of which include the individuals involving in mutation that are randomly chosen from the parent population, and thus it is beneficial to maintain the population diversity. However, DE/rand/1 is the first mutation strategy for DE [1] and is also the most widely used mutation strategy in the literature [18,20,21,24–26]. Compared to DE/rand/2, DE/rand/1 is faster and more robust [25]. Therefore, DEPSO selects DE/rand/1 strategy to improve global exploration ability in the early stage of the evolution.

Refer to Eq. (11) (PSO), the right side of which consists of three parts. The second and third parts are composed of two difference vectors. The literature [54] empirically demonstrated that the statistical distribution of two difference vectors presents a bell shape that is regarded as a better perturbation mode. This is the famous "overflying" effect that is important in exploring new regions [55]. In the bell-shaped distribution, most of the new points generated will be close to previous points. On the other hand, the infinite tails show the ability to generate outliers, which is necessary for finding more promising solutions [55]. The search strategy based on individual best solution and global best solution found so far makes PSO converge faster. Therefore, DEPSO makes more use of the improved PSO mutation strategy to enhance local search ability in the later stage of the evolution.

To further improve the performance of DE/rand/1, a new mutation strategy with elite archive strategy, named DE/e-rand/1, is proposed in DEPSO. The principle of elite archive strategy is to sort the current population $\mathbf{P}_G$ in ascending order according to individual fitness [27]. Then *SEP* superior individuals are incorporated into elite population $\mathbf{P}$, and *NP* − *SEP* remaining individuals are incorporated into non-elite population $\mathbf{Q}$. $\mathbf{P} \cup \mathbf{Q} = \mathbf{P}_G$ and $\mathbf{P} \cap \mathbf{Q} = \varnothing$ should be satisfied. In DE/e-rand/1 (refers to Eq. (13)), the individuals $\mathbf{X}_{r1,G}$ and $\mathbf{X}_{r2,G}$ are randomly chosen from elite population $\mathbf{P}$ and the individual $\mathbf{X}_{r3,G}$ is randomly chosen from non-elite population $\mathbf{Q}$. In addition, PSO is further improved to

make it suitable for the mutation operation of DEPSO. In DEPSO, a mutation individual is generated as follows:

$$\mathbf{V}_{i,G+1} = \begin{cases} \mathbf{X}^P_{r1,G} + F \cdot (\mathbf{X}^P_{r2,G} - \mathbf{X}^Q_{r3,G}) & \text{if } rand(0,1) \le SP_G \\ \omega \cdot \mathbf{X}_{i,G} + c_1 \cdot r_1 \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{i,G}) & \\ \quad + c_2 \cdot r_2 \cdot (\mathbf{X}_{gbest,G} - \mathbf{X}_{i,G}) & \text{otherwise} \end{cases} \tag{13}$$

where $rand(0, 1)$ is a uniformly distributed random number within the range [0, 1]. The selection probability for DE/e-rand/1 and PSO mutation strategies is denoted as SP_G, which is expressed as follows:

$$SP_G = \frac{1}{1 + e^{1-(G_{\max}/G+1)^{\tau}}} \tag{14}$$

where G is the current number of evolution generations and $G_{\max}$ is the maximum number of evolution generations. τ is a positive constant and its appropriate values will be given experimentally in Section 5.3.1.

To understand intuitively the mutation operation of DEPSO, the search processes of DE/e-rand/1 and PSO are plotted in Fig. 1.

From Eqs. (13) and (14), and Fig. 1, it is easy to see that the selection probability SP_G is large ($SP_G \to 1$) in the early stage of the evolution, and the probability of rand(0, 1) $\le SP_G$ is also large. DEPSO can make more use of DE/e-rand/1 mutation strategy shown in Fig. 1(a) to maintain the population diversity, and thus expand the search space as far as possible to find more promising regions and avoid premature convergence. In the later stage of the evolution, the selection probability SP_G decreases ($SP_G \overset{G\uparrow}{\to} 0.5$) as the number of iteration increases, and the probability of rand(0, 1) $\le SP_G$ gradually becomes smaller. Therefore, PSO mutation strategy shown in Fig. 1(b) has more chances to perform the mutation operation, and thus it is helpful to improve convergence speed and precision. Although DE/e-rand/1 strategy also has the chance to perform individual mutation, the introduction of elite archive strategy can ensure that DEPSO still has higher convergence speed and precision.

4.2. Elite archive strategy in DEPSO

As shown in Eq. (13) and Fig. 1(a), DE/e-rand/1 mutation strategy consists of two parts: the first part is benchmark vector $\mathbf{X}^P_{r1,G}$ chosen randomly from elite population **P**; the second part is differential vector $\mathbf{X}^P_{r2,G} - \mathbf{X}^Q_{r3,G}$. From the perspective of vector operation, the differential vector tends to the vector direction of better individual $\mathbf{X}^P_{r2,G}$, which is beneficial to improve convergence speed and precision.

Elite population **P** is mainly used to enhance convergence speed, and non-elite population **Q** is mainly used to increase population diversity. When the size of elite population is $SEP = 0$ or $SEP = NP$, the individuals $\mathbf{X}_{r1,G}$, $\mathbf{X}_{r2,G}$ and $\mathbf{X}_{r3,G}$ in DE/e-rand/1 shown Fig. 1(a) are chosen randomly from the population $\mathbf{P}_G$. Although the population diversity can be well maintained, the convergence speed will decrease. When the size of **P** is $SEP = 1$, the individuals $\mathbf{X}_{r1,G}$ in DE/e-rand/1 becomes $\mathbf{X}_{best,G}$ and the individuals $\mathbf{X}_{r2,G}$ and $\mathbf{X}_{r3,G}$ are chosen randomly from the non-elite population **Q**. Then DE/e-rand/1 mutation strategy is degenerated to DE/best/1 strategy shown in Eq. (3). Although it is beneficial to improve convergence performance, the population diversity is hard to be effectively maintained. Therefore, too large or too small SEP will affect the performance of DEPSO, and its appropriate values will be given in Section 5.3.2.

During the evolution, if the trail vector $\mathbf{U}_{i,G+1}$ is better than the corresponding target vector $\mathbf{X}_{i,G}$, **P** and **Q** will be updated. The update operation is as follows:

Step 1: Set the population size NP and the generation number G=0. Then, randomly generate an initial population $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \mathbf{X}_{2,G}, \cdots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x^1_{i,G}, x^2_{i,G}, \cdots, x^D_{i,G}\}$ ($i = 1, \cdots, NP$), where $x^j_{i,G} \in [x^j_{min}, x^j_{max}]$.
Step 2: Sort the initial population $\mathbf{P}_0$ according to each individual fitness in ascending order. Initialize the elite population **P** by choosing SEP better individuals from $\mathbf{P}_0$, and the non-elite population $\mathbf{Q} = \mathbf{P}_0 - \mathbf{P}$.
Generate randomly the control parameter values $F_{i,G}$ and $CR_{i,G}$ for each associated individual from $[F_l, F_u]$ and $[CR_l, CR_u]$, respectively.

Step 3: While G is less than the generation number G_{max}
For $i = 1$ **to** NP
Compute the selection probability of mutation strategies
$$SP_G = \frac{1}{1 + e^{1-(G_{max}/G+1)^{\tau}}}$$
Step 3.1: ***Mutation***
/*Generate a mutated vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \cdots, u^D_{i,G}\}$*/
Generate a random number $rand_i$ within [0,1].
If $rand_i \le SP_G$
/* DE/e-rand/1 mutation strategy */
$\mathbf{X}^P_{r1,G}$ and $\mathbf{X}^P_{r2,G}$ are randomly chosen from the elite population **P**, and $\mathbf{X}^Q_{r3,G}$ is chosen randomly from the non-elite population **Q**.
$\mathbf{V}_{i,G+1} = \mathbf{X}^P_{r1,G} + F_{i,G}(\mathbf{X}^P_{r2,G} - \mathbf{X}^Q_{r3,G})$
Else
/* PSO mutation strategy */
$\mathbf{V}_{i,G+1} = \omega \mathbf{X}_{i,G} + r_1 c_1 (\mathbf{X}_{pbest} - \mathbf{X}_{i,G}) + r_2 c_2 (\mathbf{X}_{gbest} - \mathbf{X}_{i,G})$
End If
Step 3.2: ***Crossover***
/*Generate a trial vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \cdots, u^D_{i,G}\}$ using binomial crossover */
For $j = 1$ **to** D
$j_{rand} = rand(1, D)$
$$u^j_{i,G+1} = \begin{cases} v^j_{i,G+1} & \text{if } rand(0,1)_j \le CR_i \,\|\, jrand = j \\ x^j_{i,G} & \text{otherwise} \end{cases} \quad j = 1, 2, \cdots, D$$
End For
Step 3.3: ***Selection***
If $f(\mathbf{U}_{i,G+1}) \le f(\mathbf{X}_{i,G})$
$\mathbf{X}_{i,G+1} = \mathbf{U}_{i,G+1}$
Set the counter for the number of individual evolution stagnation $NS_i = 0$.
Update the elite population **P** and non-elite population **Q**.
If $f(\mathbf{U}_{i,G+1}) \le f(\mathbf{X}_{pbest})$
$\mathbf{X}_{pbest} = \mathbf{U}_{i,G+1}$
End If
If $f(\mathbf{U}_{i,G+1}) \le f(\mathbf{X}_{gbest})$
$\mathbf{X}_{gbest} = \mathbf{U}_{i,G+1}$
End If
Else
$\mathbf{X}_{i,G+1} = \mathbf{X}_{i,G}$ $NS_i = NS_i + 1$
End If
Step 3.4: ***updating***
If NS_i is greater than the maximum number of individual evolution stagnation NS_{max}.
/* Update control parameter values */
$F_i = rand(F_l, F_u)$ $CR_i = rand(CR_l, CR_u)$
/* Randomly Update $\mathbf{X}_{i,G+1}$ to increase population diversity */
If $\mathbf{X}_{i,G+1}$ in the non-elite population **Q**
For $j = 1$ **to** D
$$x^j_{i,G+1} = \begin{cases} x^j_{min} + (x^j_{max} - x^j_{min}) \cdot rand(0,1) & if\ rand(0,1)_j \le \gamma \\ x^j_{i,G+1} & otherwise \end{cases}$$
End For
End If
End If
$G = G + 1$
End For
Step 4: End While

Fig. 2. Pseudo-code of DEPSO.

(1) If the corresponding target vector $\mathbf{X}_{i,G}$ is in **P**, the trail vector $\mathbf{U}_{i,G+1}$ will replace $\mathbf{X}_{i,G}$ and enter the elite population **P**.

(2) If the corresponding target vector $\mathbf{X}_{i,G}$ is not in **P** and the fitness of the trail vector $\mathbf{U}_{i,G+1}$ is less than or equal to the worst individual $\mathbf{X}^P_{worst}$ in **P**, the trail vector $\mathbf{U}_{i,G+1}$ will enter **P** and $\mathbf{X}_{i,G}$

Table 1
Test functions.

Functions	Ω	$f(*)$
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$f_2(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$	0
$f_3(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100, 100]^D$	0
$f_4(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
$f_5(x) = \max\{ \lvert x_i \rvert , 1 \le i \le D \}$	$[-100, 100]^D$	0
$f_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0
$f_7(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^D$	0
$f_8(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	$[-100, 100]^D$	0
$f_9(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^D$	0
$f_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos 2\pi x_i) + 20 + e$	$[-32, 32]^D$	0
$f_{11}(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10 \cos(2\pi x_i))$	$[-0.5, 0.5]^D$	0
$f_{12}(x) = \sum_{i=1}^{D} (\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)]$ $a = 0.5, b = 3, k_{\max} = 20$	$[-0.5, 0.5]^D$	0
$f_{13}(x) = \sum_{i=1}^{D} (0.5 + \frac{\sin 2(\sqrt{x_i^2 + x_{i+1}^2}) - 0.5}{(1 + 0.001(x_i^2 + x_{i+1}^2))^2}) x_{D+1} = x_1$	$[-0.5, 0.5]^D$	0
$f_{14}(x) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2}$	$[-100, 100]^D$	0
$f_{15}(x) = \frac{\pi}{D} \left\{ 10 \sin 2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin 2(\pi y_{i+1})] + (y_D - 1)^2 \right\}$ $+ \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^D$	0
$f_{16}(x) = 0.1 \left\{ 10 \sin 2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin 2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin 2(2\pi x_D)] \right\}$ $+ \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$	0

is needed to be removed from **Q**. In addition, $\mathbf{X}_{worst}^{P}$ removed from **P** should be added to **Q**.

4.3. Control parameters adaptation

In the conventional DE, the control parameters are applied at the population level, and thus each individual in the population has the same parameter settings throughout the evolution process. However, different parameter settings for each individual may achieve better performance during different stages of the evolution. In addition, the conventional DE employs a simple fixed parameter settings method, while the appropriate parameter values are difficult to be determined. For optimization problems with different characteristics, the best parameter settings are often different. Control parameters adaptation strategy can effectively address the aforementioned problems and improve the applicability of DE. Therefore, DEPSO introduces a parameters adaptation strategy similar to [27] to ensure that the

Table 2
Mean and STD obtained by the conventional DE, PSO and DEPSO on 30D functions.

Functions	Gen	Mean(STD)						
		DE-P1	DE-P2	DE-P3	DE-P4	PSO-P1	PSO-P2	DEPSO
f_1	1000	3.25E−11(6.30E−12)−	5.69E−08(3.05E−08)−	1.35E−03(3.31E−04)−	1.12E+04(1.66E+003)−	1.46E−04(2.33E−04)−	1.24E+00(7.41E−01)−	2.34E−102(1.26E−101)
f_2	1000	1.10E+04(1.98E+03)−	2.61E+01(1.33E+01)−	1.47E+04(1.61E+03)−	2.82E+04(2.61E+03)−	3.71E+01(1.21E+01)−	1.32E+02(3.24E+01)−	6.79E−70(3.65E−69)
f_3	1000	6.57E−08(1.90E−08)−	2.70E−05(1.82E−05)−	1.07E+00(2.17E−01)−	2.77E+06(4.60E+05)−	1.08E+00(1.35E+00)−	1.03E+04(8.03E+03)−	2.25E−99(1.21E−98)
f_4	1000	2.75E−07(5.18E−08)−	2.34E−04(7.65E−05)−	3.75E−03(5.15E−04)−	6.20E+01(5.37E+00)−	8.37E−04(4.27E−04)−	2.72E−01(7.91E−02)−	4.55E−57(1.91E−56)
f_5	2000	1.18E−02(1.49E−03)−	6.85E−02(1.39E−01)−	2.59E+00(2.25E−01)−	4.49E+01(4.29E+00)−	3.95E−01(1.22E−01)−	1.72E+00(2.76E−01)−	2.99E−88(1.37E−87)
f_6	500	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)≈	9.03E+00(1.08E+00)−	1.76E+04(1.72E+03)−	7.18E+01(4.84E+01)−	9.21E+01(1.18E+02)−	0.0E+00(0.0E+00)
f_7	2000	1.32E−02(2.64E−03)−	6.91E−03(1.67E−03)−	3.04E−02(4.82E−03)−	1.33E+00(6.39E−01)−	2.09E−03(5.93E−04)−	2.98E−03(8.06E−04)−	1.85E−03(1.29E−03)
f_8	2000	3.45E+01(1.07E+01)−	1.43E+01(1.02E+01)−	6.49E+01(1.23E+01)−	2.72E+08(1.75E+08)−	5.90E+01(5.91E+01)−	1.94E+02(1.06E+02)−	5.76E−01(1.36E+00)
f_9	1000	9.93E−10(1.94E−09)+	2.47E−04(1.33E−03)+	1.52E−02(4.19E−03)−	9.60E+01(2.21E+01)−	8.47E−03(9.14E−03)−	8.15E−01(1.72E−01)−	1.40E−03(3.16E−03)
f_{10}	1000	1.53E−06(1.66E−07)−	6.43E−05(1.41E−05)−	1.01E−02(1.10E−03)−	1.58E+01(7.85E−01)−	2.02E−03(1.46E−03)−	3.96E−01(1.95E−01)−	4.00E−15(0.0E+00)
f_{11}	1000	1.52E−13(4.31E−14)−	1.57E−10(7.62E−011)−	5.92E−06(1.05E−06)−	4.71E+01(8.69E+00)−	5.70E−07(5.01E−07)−	5.67E−03(3.12E−03)−	0.0E+00(0.0E+00)
f_{12}	500	1.12E−01(1.02E−02)−	1.75E+00(2.68E−01)−	2.08E+00(1.64E−01)−	3.41E+01(1.12E+00)−	9.38E−01(5.97E−01)−	3.46E+00(7.65E−01)−	0.0E+00(0.0E+00)
f_{13}	500	6.27E−08(1.58E−08)−	1.68E−06(6.20E−07)−	3.37E−04(4.56E−05)−	8.48E−01(1.27E−01)−	5.83E−06(5.91E−06)−	6.59E−04(4.00E−04)−	0.0E+00(0.0E+00)
f_{14}	500	1.26E+00(9.98E−02)−	8.32E−01(1.01E−01)−	3.28E+00(1.99E−01)−	1.42E+01(9.25E−01)−	5.24E−01(5.59E−02)−	8.97E−01(1.08E−01)−	9.99E−02(1.93E−05)
f_{15}	1000	9.20E−13(2.68E−13)−	3.00E−09(2.21E−09)−	7.67E−05(2.48E−05)−	9.41E+06(4.52E+06)−	2.86E−07(2.45E−07)−	3.49E−03(3.06E−03)−	1.57E−32(8.21E−48)
f_{16}	1000	6.20E−12(1.85E−12)−	2.31E−08(1.62E−08)−	3.25E−04(9.73E−05)−	3.39E+07(1.34E+07)−	1.15E−03(3.29E−03)−	7.54E−02(3.33E−02)−	1.35E−32(5.47E−48)
−		14	14	16	16	16	16	
≈		1	1	0	0	0	0	
+		1	1	0	0	0	0	

Table 3
SR and MNEG obtained by the conventional DE, PSO and DEPSO on 30D functions.

Functions	DE-P1		DE-P2		DE-P3		DE-P4		PSO-P1		PSO-P2		DEPSO	
	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG
f_1	100	836	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**198**
f_2	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**403**
f_3	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**248**
f_4	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**272**
f_5	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**770**
f_6	100	298	100	385	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**91**
f_7	13	N/A	93	N/A	0	N/A	0	N/A	**100**	**729**	100	914	100	1155
f_8	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**83**	N/A
f_9	**97**	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	83	N/A
f_{10}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**298**
f_{11}	100	681	100	842	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**164**
f_{12}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**289**
f_{13}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**136**
f_{14}	0	N/A	93	N/A	0	N/A	0	N/A	100	335	0	N/A	**100**	**107**
f_{15}	100	734	100	947	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**214**
f_{16}	100	786	30	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**210**

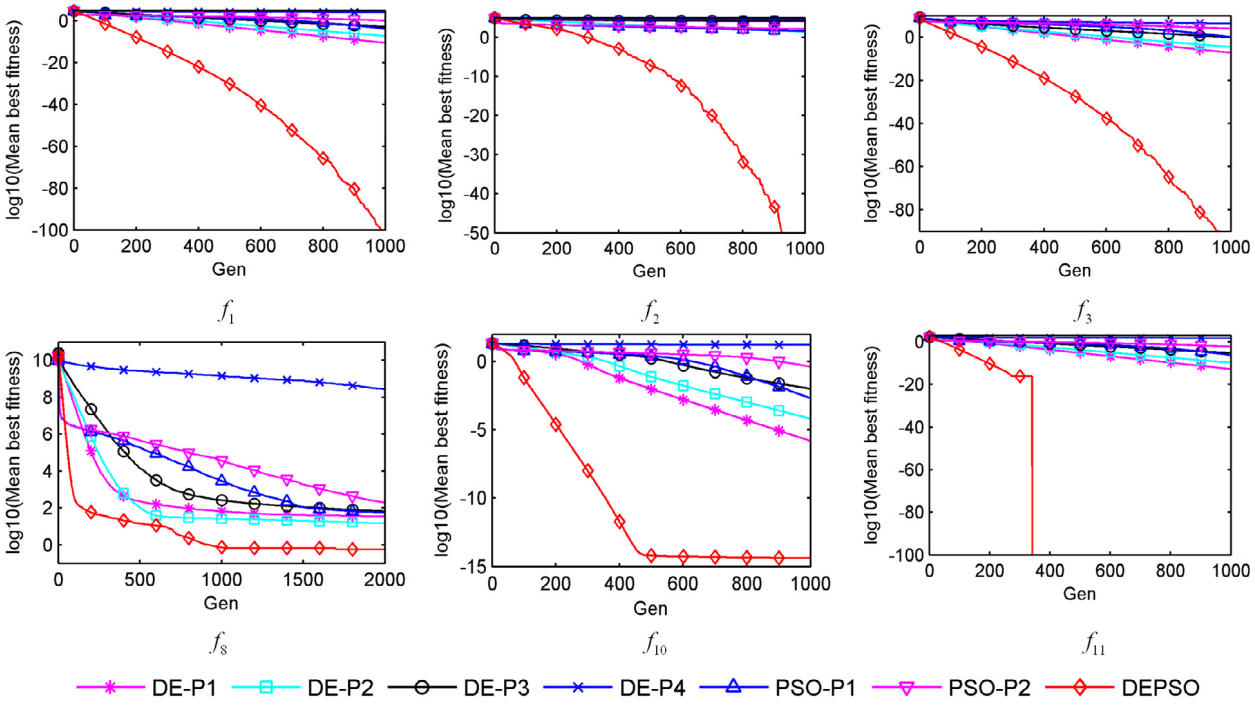

Fig. 3. Evolution curves of the conventional DE, PSO and DEPSO on 30D functions.

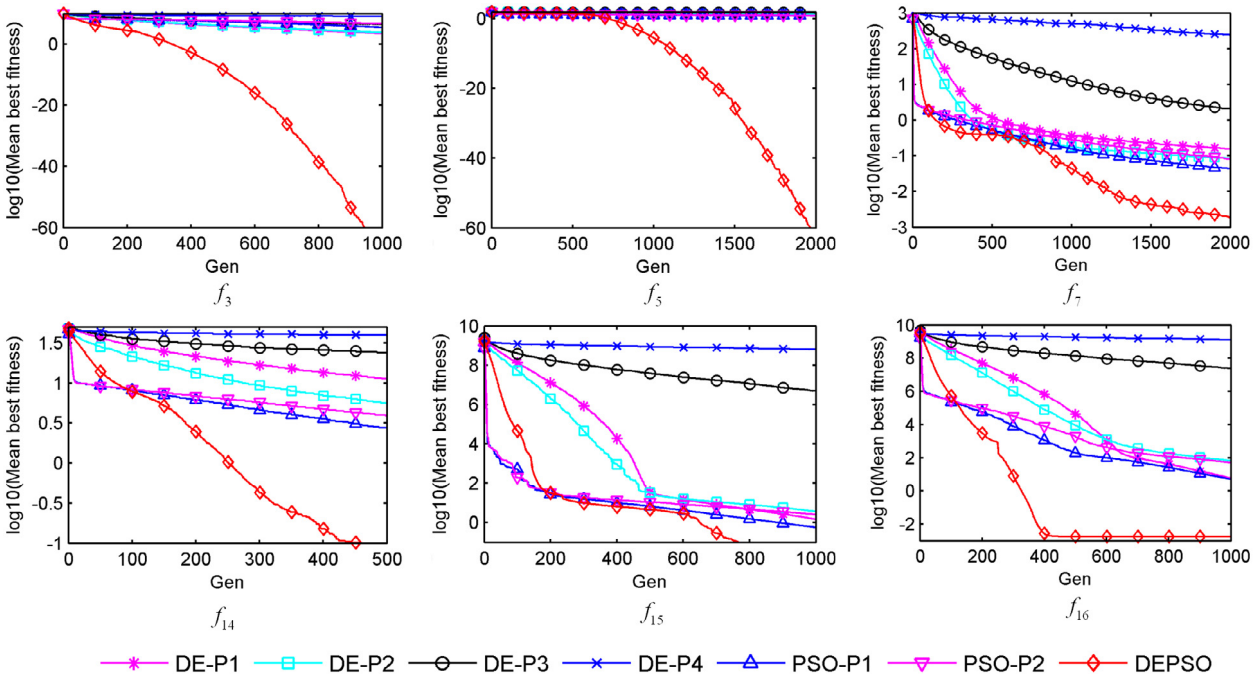

Fig. 4. Evolution curves of the conventional DE, PSO and DEPSO on 100D functions.

Table 4
The results of Friedman and Kruskal–Wallis tests for the conventional DE, PSO and DEPSO on 30D functions.

Algorithms	DE-P1	DE-P2	DE-P3	DE-P4	PSO-P1	PSO-P2	DEPSO
Friedman (Rank)	2.56	2.88	5.06	7.00	3.81	5.50	**1.19**
Kruskal–Wallis (Rank)	40.09	43.72	63.81	97.38	57.56	73.00	**19.94**

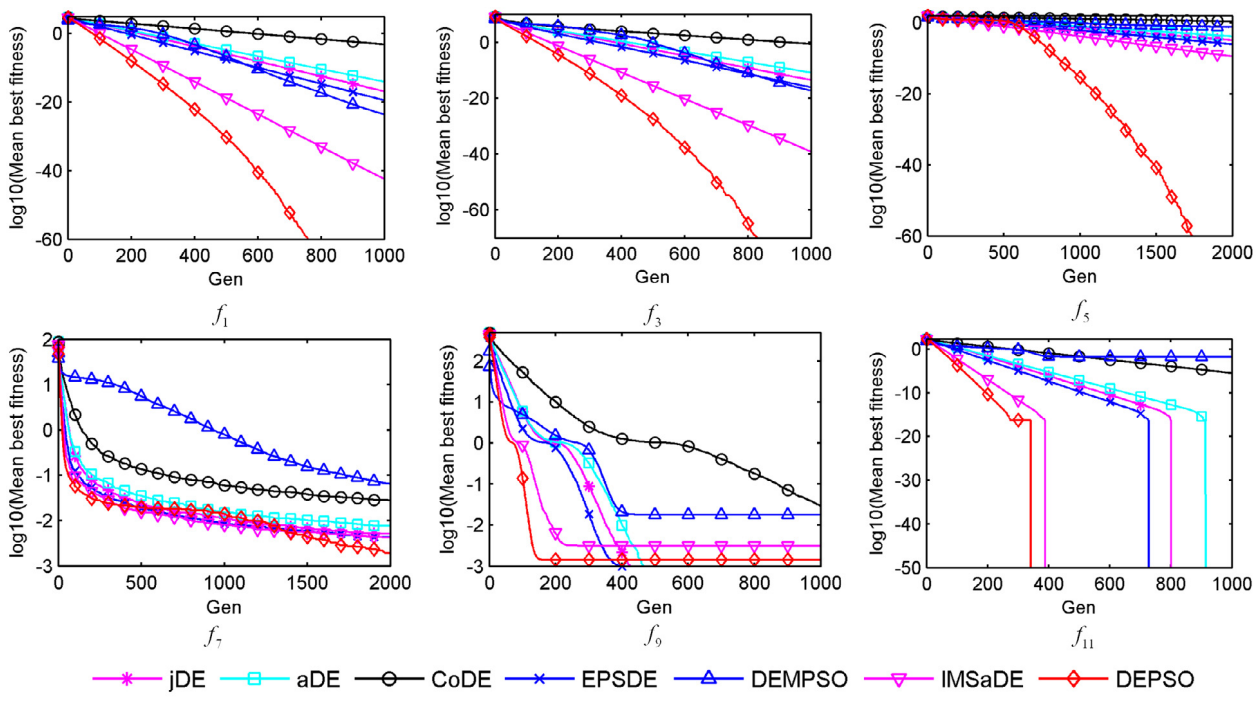

Fig. 5. Evolution curves of jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 30D functions.

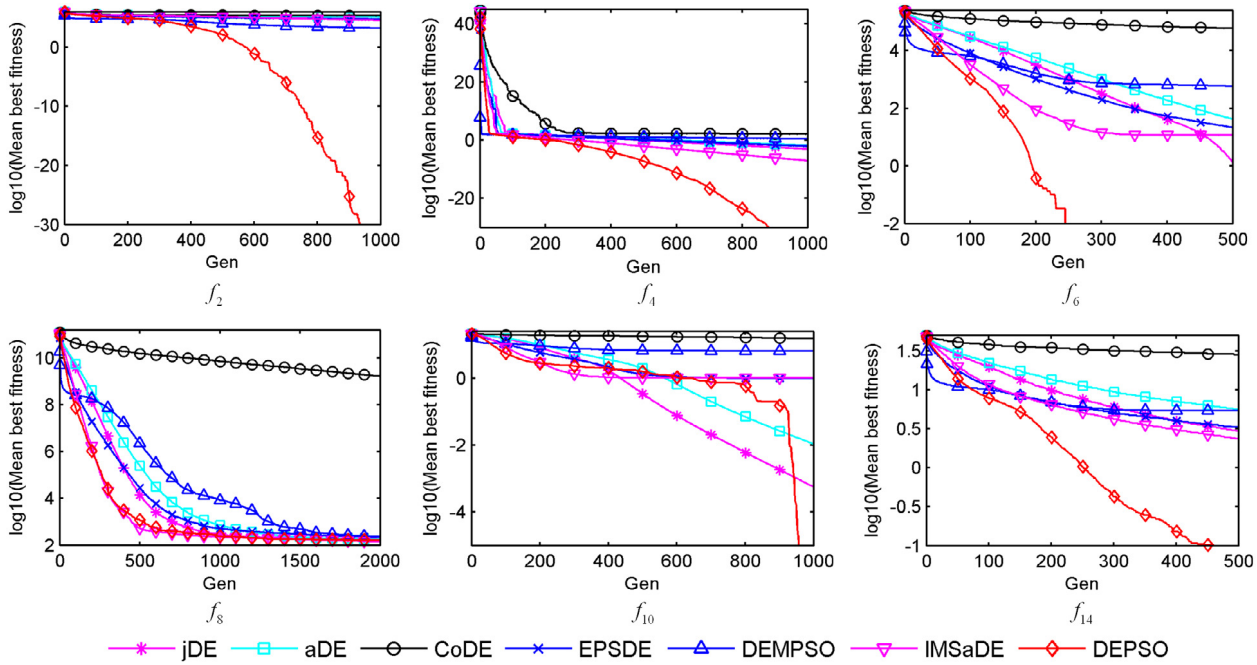

Fig. 6. Evolution curves of jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 100D functions.

Table 5
Mean and STD obtained by the conventional DE, PSO and DEPSO on 100D functions.

Functions	Gen	Mean(STD)						
		DE-P1	DE-P2	DE-P3	DE-P4	PSO-P1	PSO-P2	DEPSO
f_1	1000	2.83E+00(3.34E−01)−	4.66E+00(1.69E+00)−	1.03E+04(7.55E+02)−	1.16E+05(1.15E+04)−	3.70E+01(1.01E+01)−	3.31E+02(4.98E+01)−	2.75E−73(1.08E−72)
f_2	1000	2.06E+05(1.56E+04)−	1.42E+05(2.58E+04)−	2.26E+05(2.23E+04)−	3.39E+05(4.16E+04)−	1.08E+04(2.49E+03)−	1.43E+04(2.89E+03)−	1.62E−42(8.71E−42)
f_3	1000	3.11E+03(2.94E+02)−	6.45E+03(1.88E+03)−	3.16E+06(1.87E+05)−	1.21E+09(1.65E+08)−	3.13E+05(1.31E+05)−	4.93E+06(1.48E+06)−	3.83E−71(1.89E−70)
f_4	1000	6.04E−01(4.25E−02)−	1.64E+00(3.38E−01)−	6.02E+01(3.19E+00)−	6.31E+25(2.78E+26)−	2.68E+00(4.58E−01)−	1.26E+01(1.84E+00)−	6.57E−40(3.40E−39)
f_5	2000	2.16E+01(6.39E−01)−	1.92E+01(4.11E+00)−	5.55E+01(1.03E+00)−	8.27E+01(1.61E+00)−	7.25E+00(6.00E−01)−	9.25E+00(7.57E−01)−	2.54E−65(1.02E−64)
f_6	500	7.13E+02(6.90E+01)−	6.28E+02(1.43E+02)−	3.23E+04(2.16E+03)−	1.44E+05(1.02E+04)−	6.17E+02(6.53E+02)−	1.21E+03(7.04E+02)−	0.0E+00(0.0E+00)
f_7	2000	1.60E−01(1.92E−02)−	8.47E−02(2.07E−02)−	2.07E+00(2.04E−01)−	2.42E+02(5.08E+01)−	4.31E−02(8.38E−03)−	8.14E−02(1.09E−02)−	1.86E−03(9.15E−04)
f_8	2000	5.59E+02(6.37E+01)−	3.35E+02(2.04E+02)−	7.96E+07(1.12E+07)−	2.10E+10(5.07E+09)−	3.65E+03(1.38E+03)−	2.28E+05(9.13E+04)−	1.63E+02(5.09E+01)
f_9	1000	8.32E−01(3.81E−02)−	9.10E−01(1.09E−01)−	9.21E+01(5.68E+00)−	1.07E+03(9.83E+01)−	1.29E+00(6.92E−02)−	4.05E+00(6.77E−01)−	1.81E−02(3.31E−02)
f_{10}	1000	3.86E−01(3.84E−02)−	6.11E−01(2.07E−01)−	1.16E+01(2.20E−01)−	1.96E+01(3.04E−01)−	2.05E+00(2.37E−01)−	3.99E+00(1.83E−01)−	1.79E−11(9.64E−11)
f_{11}	1000	1.07E−02(1.36E−03)−	1.83E−02(7.69E−03)−	3.70E+01(2.26E+00)−	4.55E+02(3.78E+01)−	1.80E−01(4.81E−02)−	1.65E+00(2.66E−01)−	0.0E+00(0.0E+00)
f_{12}	500	2.84E+01(9.32E−01)−	3.61E+01(3.44E+00)−	9.05E+01(2.11E+00)−	1.49E+02(3.73E+00)−	2.42E+01(2.51E+00)−	3.70E+01(2.82E+00)−	0.0E+00(0.0E+00)
f_{13}	500	3.44E−02(2.77E−03)−	2.93E−02(7.74E−03)−	1.56E+00(1.01E−01)−	7.01E+00(5.08E−01)−	8.63E−03(1.78E−03)−	3.89E−02(6.19E−03)−	0.0E+00(0.0E+00)
f_{14}	500	1.12E+01(3.33E−01)−	5.61E+00(5.10E−01)−	2.40E+01(5.80E−01)−	3.96E+01(1.16E+00)−	2.73E+00(1.73E−01)−	3.92E+00(1.78E−01)−	1.00E−01(5.46E−04)
f_{15}	1000	1.47E+00(2.16E−01)−	3.65E+00(2.32E+00)−	4.88E+06(1.23E+06)−	6.28E+08(1.54E+08)−	5.68E−01(3.93E−01)−	2.56E+00(1.27E+00)−	1.04E−02(2.45E−02)
f_{16}	1000	5.76E+00(7.67E−01)−	6.72E+01(8.02E+01)−	2.41E+07(3.96E+06)−	1.28E+09(2.91E+08)−	4.94E+00(1.69E+00)−	5.04E+01(2.01E+01)−	1.83E−03(4.09E−03)
−		16	16	16	16	16	16	
≈		0	0	0	0	0	0	
+		0	0	0	0	0	0	

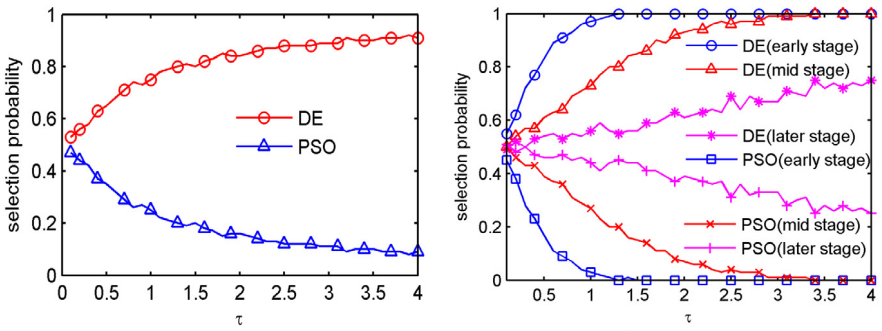

Fig. 7. Selection probability of mutation strategies in DEPSO.

Table 6
SR and MNEG obtained by the conventional DE, PSO and DEPSO on 100D functions.

Functions	DE-P1		DE-P2		DE-P3		DE-P4		PSO-P1		PSO-P2		DEPSO	
	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG
f_1	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**432**
f_2	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**679**
f_3	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**486**
f_4	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**514**
f_5	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**1070**
f_6	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**199**
f_7	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**1224**
f_8	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
f_9	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**57**	N/A
f_{10}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**760**
f_{11}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**389**
f_{12}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**389**
f_{13}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**250**
f_{14}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**100**	**252**
f_{15}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**80**	N/A
f_{16}	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	**83**	N/A

Table 7
The results of Friedman and Kruskal–Wallis tests for the conventional DE, PSO and DEPSO on 100D functions.

Algorithms	DE-P1	DE-P2	DE-P3	DE-P4	PSO-P1	PSO-P2	DEPSO
Friedman (Rank)	3.25	3.50	5.94	7.00	2.88	4.44	**1.00**
Kruskal–Wallis (Rank)	50.31	52.62	78.19	88.56	52.31	60.44	**13.06**

selection of parameter values is independent of the specific problem and meet the needs of each individual during the different stages of the evolution.

During the evolution, the parameter settings tend to produce better individuals that are more likely to survive and enter the next generation; otherwise these parameter values need to be reset to avoid evolutionary stagnation. However, when to adjust the parameter values is often difficult. In DEPSO, the parameter values can be adjusted by tracking the evolutionary status of each individual in real time. If any individual in the current population fails to produce a better offspring for consecutive NS_{max} generations, it can be considered that the individual evolution is stagnant and its corresponding parameter values need to be readjusted. In addition, there is no uniform regulation for adjusting the parameter values, while a random adjustment strategy is generally considered a good choice [21]. Base on the above analysis, the following parameter adaptation strategy can be obtained.

$$F_{i,G+1} = \begin{cases} F_{i,G} & \text{if } NS_i < NS_{max} \\ F_l + rand(0,1)\cdot(F_u - F_l) & \text{otherwise} \end{cases} \quad (15)$$

$$CR_{i,G+1} = \begin{cases} CR_{i,G} & \text{if } NS_i < NS_{max} \\ CR_l + rand(0,1)\cdot(CR_u - CR_l) & \text{otherwise} \end{cases} \quad (16)$$

where $F_{i,G}$ and $CR_{i,G}$ are the scaling factor and crossover rate of the target individual $\mathbf{X}_{i,G}$ at generation G, respectively. $F_l = 0.1$ and $F_u = 0.8$ are the lower and upper bounds of the scaling factor, respectively. $CR_l = 0.3$ and $CR_u = 1.0$ are the lower and upper bounds of the crossover rate. $rand(0,1)$ is a uniform random number in the range $[0, 1]$. NS_i is a counter for the number of individual stagnation generations. If the target individual $\mathbf{X}_{i,G}$ can generate a better offspring, then let $NS_i = 0$. $NS_{max} = 5$ is the maximum number of individual stagnation generations. The parameter adaptation can retain the current parameter values for consecutive NS_{max} generations, and thus is able to search more promising regions as far as possible and improve convergence precision.

To increase population diversity and avoid premature convergence, DEPSO utilizes Eq. (17) to adjust the individual $\mathbf{X}_{i,G}$ if it is in the non-elite population $\mathbf{Q}$ and $NS_i \geq NS_{max}$.

$$x^j_{i,G} = \begin{cases} X^j_{min} + rand(0,1)\cdot(X^j_{max} - X^j_{min}) \\ \quad \text{if } rand(0,1)_j \leq \gamma \\ x^j_{i,G} \quad \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, D \quad (17)$$

where $rand(0,1)_j$ is a random number within the range $[0, 1]$. γ is a adjustment probability and should not be too large. Otherwise, more new elements are added to $\mathbf{X}_{i,G}$ that may cause the individual to deviate from the correct evolutionary direction, thus

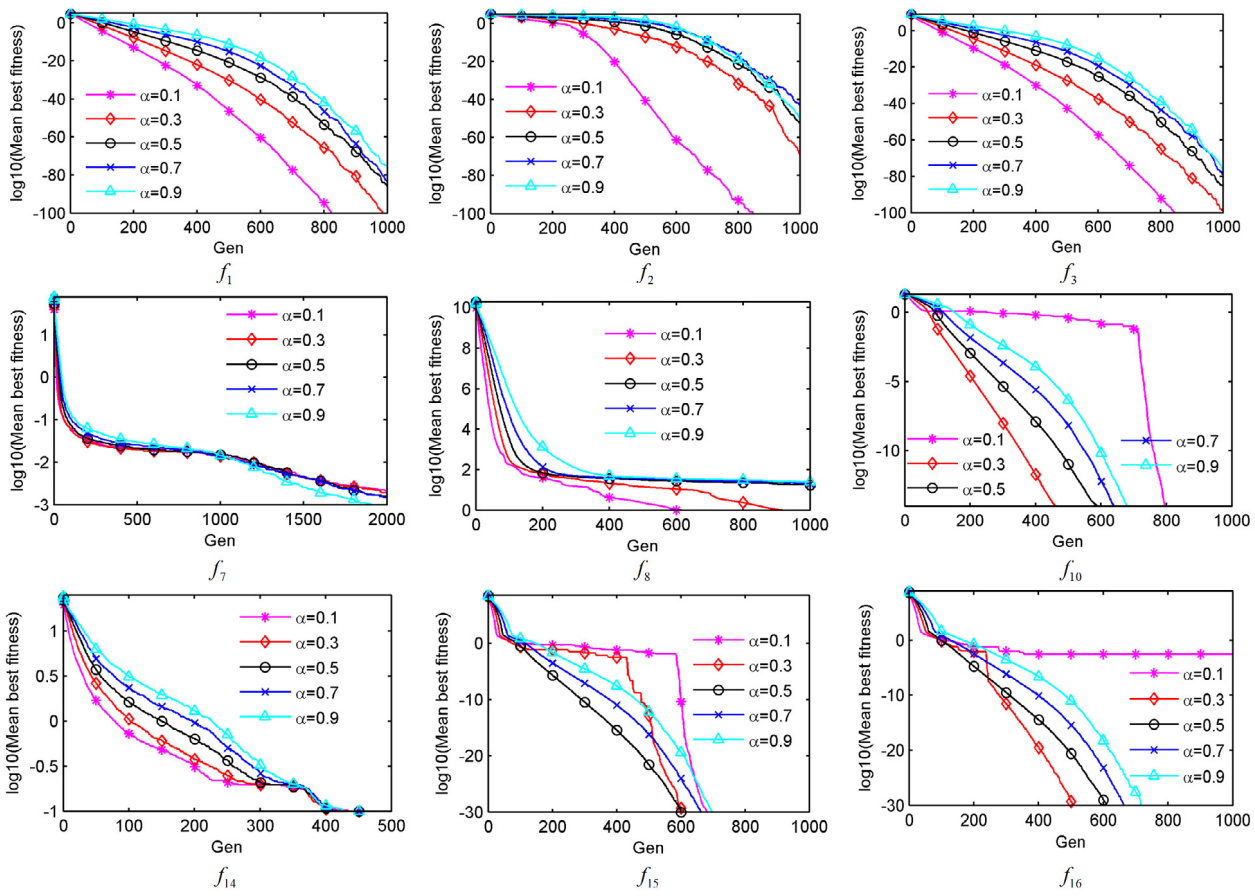

Fig. 8. Evolution curves obtained by DEPSO with different size of elite population.

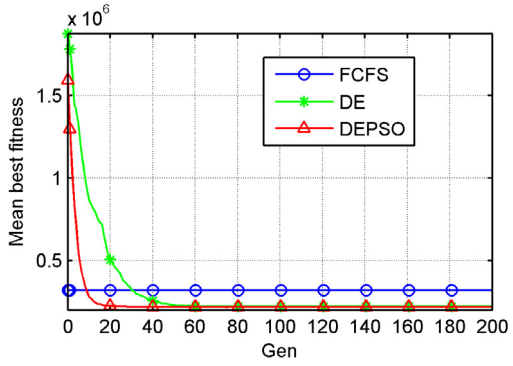

Fig. 9. Evolution curves of FCFS, DE and DEPSO.

decreasing convergence speed and precision. In the paper, γ is set to $\gamma = 0.001$.

The pseudo code of DEPSO is shown in Fig. 2.

5. Experimental study and analysis

Test functions shown in Table 1 are used to study the optimization performance of DEPSO. $f_1 \sim f_6$ are unimodal functions and f_7 is a noisy quadratic function. $f_8 \sim f_{16}$ are multimodal functions and the number of their local minima increases exponentially as the dimension increases. $f(*)$ is the global minimum. The dimensions for all functions are set to $D = 30,100$, and 30 independent runs on each function for each algorithm are performed. For clearness, the Mean and standard deviation (STD) of the function error value between the minimum obtained and $f(*)$ are recorded. In addition, the mean number of evolution generations (MENG) to reach the specified error precision and success rate (SR) of each algorithm are summarized. If SR is not 100%, MENG is denoted as N/A (Not Applicable). Three statistical tests with a significance level of 5%: Wilcoxon's rank sum test, Friedman's test and Kruskal–Wallis's test [56] are used to further compare the performances of all algorithms. For clarity, the results of the best and second best algorithms are shown in bold and italic respectively, if needed. The compared algorithms are implemented using Visual Studio 2008 on Windows 7 operating system according to the corresponding literature.

5.1. Comparison with the conventional DE and PSO

In this experiment, the proposed DEPSO is compared with the conventional DE and PSO on 30-dimensional and 100-dimensional test functions. The error precisions for test functions are set as follows: 1.0^{-2} for f_7, 1.0 for f_{14}, and 1.0^{-8} for all others, as used in the literature [23]. For each algorithm, the population size is set to $NP = 100$. For DEPSO, the size of elite population is set to $SEP = 30$. For DE, scaling factor F is set to $F \in 0.5, 0.9$ and crossover rate CR is set to $CR \in 0.1, 0.9$. For PSO, inertia weight ω is set to $\omega_{\max} = 0.9$ and $\omega_{\min} \in 0.4, 0.6$. In addition, the comparative experiments are carried out by using the design of experiments (DOE). Therefore, the parameter settings are as follows: DE-P1 ($F = 0.5$, $CR = 0.1$), DE-P2 ($F = 0.5$, $CR = 0.9$), DE-P3 ($F = 0.9$, $CR = 0.1$), DE-P4 ($F = 0.9$, $CR = 0.9$), PSO-P1 ($\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$), PSO-P2 ($\omega_{\max} = 0.9$, $\omega_{\min} = 0.6$).

Table 8
Experimental results of jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 30D functions.

Functions	Gen	Mean(STD)						
		jDE	aDE	CoDE	EPSDE	DEMPSO	IMSaDE	DEPSO
f_1	1000	1.49E−17(1.53E−17)−	9.23E−15(4.19E−15)−	8.04E−04(1.74E−04)−	3.56E−20(4.14E−20)−	2.75E−24(5.22E−24)−	3.67E−43(4.80E−43)−	2.34E−102(1.26E−101)
f_2	1000	1.14E+01(1.22E+01)−	4.60E+02(1.59E+02)−	1.55E+04(2.17E+03)−	8.48E+01(4.52E+01)−	6.26E−06(8.87E−06)−	9.39E−02(8.62E−02)−	6.79E−70(3.65E−69)
f_3	1000	3.06E−14(2.84E−14)−	1.42E−11(5.95E−12)−	2.57E−01(5.29E−02)−	7.54E−17(6.54E−17)−	6.00E−18(1.91E−17)−	6.69E−40(1.74E−39)−	2.25E−99(1.21E−98)
f_4	1000	4.22E−11(2.15E−11)−	2.11E−09(6.90E−10)−	2.81E−03(3.68E−04)−	3.62E−10(1.70E−10)−	1.97E−04(5.30E−04)−	5.27E−24(4.57E−24)−	4.55E−57(1.91E−56)
f_5	2000	1.24E−05(3.73E−06)−	1.41E−04(3.44E−05)−	2.44E+00(1.80E−01)−	9.57E−07(9.73E−07)−	6.51E−02(5.92E−02)−	3.42E−10(4.45E−10)−	2.99E−88(1.37E−87)
f_6	500	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)≈	7.17E+00(1.39E+00)−	0.0E+00(0.0E+00)≈	2.00E+00(1.65E+00)−	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)
f_7	2000	5.08E−03(1.17E−03)−	7.63E−03(1.94E−03)−	2.76E−02(5.40E−03)−	4.31E−03(1.16E−03)−	6.14E−03(3.10E−03)−	4.29E−03(9.66E−04)−	1.85E−03(1.29E−03)
f_8	2000	2.02E+01(1.44E+01)−	2.06E+01(1.01E+01)−	5.60E+01(1.34E+01)−	1.91E+00(1.76E+00)−	9.30E−01(1.69E+00)−	5.32E−01(1.36E+00)+	5.76E−01(1.36E+00)
f_9	1000	2.81E−16(8.50E−16)+	1.17E−13(3.78E−13)+	2.91E−02(6.98E−03)−	9.04E−04(2.81E−03)+	1.83E−02(1.91E−02)−	3.12E−03(5.79E−03)−	1.40E−03(3.16E−03)
f_{10}	1000	7.70E−10(3.23E−10)−	2.84E−08(7.51E−09)−	8.18E−03(8.50E−04)−	5.67E−11(3.01E−11)−	3.05E+00(6.44E−01)−	6.48E−15(1.63E−15)−	4.00E−15(0.0E+00)
f_{11}	1000	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)≈	3.27E−06(7.68E−07)−	0.0E+00(0.0E+00)≈	1.66E−15(1.65E−15)−	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)
f_{12}	500	9.21E−03(2.48E−03)−	2.84E−02(3.72E−03)−	2.30E+00(1.85E−01)−	8.20E−02(4.09E−02)−	6.34E+00(1.04E+00)−	2.23E−08(2.37E−08)−	0.0E+00(0.0E+00)
f_{13}	500	4.71E−11(1.84E−11)−	9.10E−10(2.85E−10)−	2.58E−04(3.92E−05)−	1.53E−12(8.32E−13)−	5.78E−15(5.98E−15)−	0.0E+00(0.0E+00)≈	0.0E+00(0.0E+00)
f_{14}	500	4.61E−01(5.66E−02)−	7.18E−01(6.80E−02)−	2.89E+00(1.58E−01)−	3.86E−01(5.14E−02)−	8.63E−01(1.35E−01)−	2.93E−01(2.06E−02)−	9.99E−02(1.93E−05)
f_{15}	1000	8.38E−19(7.26E−19)−	5.79E−16(4.10E−16)−	3.09E−04(1.15E−04)−	1.20E−20(2.16E−20)−	2.56E−01(3.87E−01)−	1.57E−32(8.21E−48)≈	1.57E−32(8.21E−48)
f_{16}	1000	6.53E−18(6.43E−18)−	4.27E−15(2.02E−15)−	6.65E−04(1.60E−04)−	5.59E−20(1.26E−19)−	4.44E−02(1.27E−01)−	1.35E−32(5.47E−48)≈	1.35E−32(5.47E−48)
−		13	13	16	13	16	10	
≈		2	2	0	2	0	5	
+		1	1	0	1	0	1	

Table 9
The results of Friedman and Kruskal–Wallis tests for jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 30D functions.

Algorithms	jDE	aDE	CoDE	EPSDE	DEMPSO	IMSaDE	DEPSO
Friedman (Rank)	3.81	4.88	6.75	3.56	5.19	2.22	**1.59**
Kruskal–Wallis (Rank)	53.19	57.25	83.94	53.62	72.62	43.06	**31.81**

5.1.1. 30 dimensional problems

Mean and STD of 30-dimensional functions obtained by each algorithm are listed in Table 2, in which Wilcoxon's rank sum test is performed. "-", "+" and "≈" indicate that the performance of the proposed DEPSO is better than, worse than, and similar to that of the compared algorithm, respectively. "Gen" denotes the number of iterations for each algorithm. SR and MNEG obtained are listed in Table 3. Additionally, the results of Friedman's test and Kruskal–Wallis's test are shown in Table 4, and the evolution curves for test functions are plotted in Fig. 3.

For unimodal functions $f_1 \sim f_5$, DEPSO is the best, while DE-P2, DE-P3, DE-P4, PSO-P1 and PSO-P2 cannot reach the specified error precision within the limited number of evolution generations. Although DE-P1, DE-P2 and DEPSO can find the global minimum for unimodal function f_6, the smaller MNEG obtained by DEPSO indicates that its convergence speed is faster. For noisy quadratic function f_7, although DEPSO obtains the smallest Mean and STD, its convergence speed is slower than those of PSO-P1 and PSO-P2. However, DE with different parameter settings cannot achieve 100% success rate.

For multimodal functions f_8, the SR obtained by DEPSO is significantly higher than the compared algorithms. For f_9, DE-P1 obtains the best results. For $f_{10} \sim f_{16}$, DEPSO obtains the best Mean and STD among the seven methods. Furthermore, the SR and MNEG obtained on these functions also indicate that it has higher the reliability and convergence speed.

From Table 3, it can be observed that the performance of DE-P1 on 30-dimensional test problems is slightly better than DE-P2 but significantly better than DE-P3 and DE-P4. However, DE-P2 outperforms DE-P1 on f_7 and f_{14}. It indicates that the performance of DE is sensitive to its parameter settings. Therefore, choosing appropriate parameter values is generally time-consuming and fixed parameter settings is not beneficial to improve the generality of the algorithm. In addition, the performance of PSO-P1 is significantly better than PSO-P2.

Overall, the performance of DEPSO is the best. This can be because DEPSO could effectively balance global exploration in the early stage and local exploitation in the later stage and thus has higher convergence speed and robustness. However, DE usually converges slowly within a limited number of evolution generations and is sensitive to its parameter settings. Although PSO has faster convergence speed in the early stage, the rapidly reduced population diversity is more likely to lead to premature convergence.

In summary, the performance of DEPSO is significantly better than the conventional DE and PSO on 30 dimensional problems shown in Table 1.

5.1.2. 100 dimensional problems

The results of 100 dimensional functions for DEPSO, DE, and PSO are summarized in Tables 5 and 6. The results of Friedman's test and Kruskal–Wallis's test are presented in Table 7, and the evolution curves are plotted in Fig. 4.

(1) In terms of convergence precision (Mean and STD), DEPSO is the best among the seven algorithms. DE-P1, DE-P2, DE-P3, DE-P4, PSO-P1 and PSO-P2 cannot be significantly better than DEPSO on any test problems. The outstanding performance of DEPSO should be due to its self-adaptive mutation strategy and elite archive strategy, which leads to good search diversity and faster convergence speed. However, DE usually converges very slowly while PSO easily suffers from premature convergence.

(2) In terms of convergence speed (MENG) and the reliability (SR), the SRs of the conventional DE and PSO on all test problems are 0, while DEPSO achieves 100% success rate on $f_1 \sim f_7$ and $f_{10} \sim f_{14}$. Even for multimodal functions f_9, f_{15} and f_{16}, DEPSO is also able to reach the specified error precision in 30 independent runs. In contrast, DEPSO still has faster convergence speed and higher robustness.

Moreover, it is clear from Table 7 that the DEPSO is significantly better than DE and PSO with respect to 100 dimensional problems. The clear difference can be observed as the dimension of the problem increases from 30D to 100D.

5.2. Comparison with the other optimization algorithms

5.2.1. Comparison with 5 improved DE on 30-dimensional problems

In this section, DEPSO is compared with jDE [20], aDE [21], CoDE [24], EPSDE [25], DEMPSO [31] and IMSaDE [27] on 30 dimensional functions. The population size for each algorithm is set to $NP = 100$. For DEPSO, the size of elite population is set to $SEP = 30$. The parameter settings of the other algorithms are the same as in the corresponding literature. The Mean and STD obtained are listed in Table 8, and the statistical analysis results for Friedman's test and Kruskal–Wallis's test are shown in Table 9.

It can be observed from Table 8 that both jDE and aDE perform worse than DEPSO on 13 problems. This is because both algorithms adopt the DE/rand/1 mutation strategy and usually converge slowly within the limited number of evolution generations. Unfortunately, CoDE and DEMPSO cannot be significantly better than DEPSO on any problems because the two algorithms converges very slowly and thus the solutions obtained are poor. EPSDE is inferior to DEPSO on 13 problems because it uses the greedy strategy "DE/best/2" and thus is more likely to suffer from premature convergence. IMSaDE utilizes the improved "DE/rand/2" strategy to improve the population diversity and convergence speed and thus outperforms jDE, aDE, CoDE, DEMPSO and EPSDE on most test problems. However, it is better than DEPSO on only one problem. In contrast, the proposed DEPSO obtains better performance on most test problems. This is due to the beneficial cooperation between DE/e-rand/1 mutation strategy and PSO mutation strategy.

Moreover, it can be seen from Table 9 that the average performance of DEPSO is significantly better than the other compared DE algorithms on 30 dimensional problems. The evolution curves derived from DEPSO, jDE, aDE, CoDE, EPSDE, DEMPSO and IMSaDE are plotted in Fig. 5 for some test problems.

5.2.2. Comparison with 5 improved DE on 100-dimensional problems

In this section, 100-dimensional functions are used to evaluate the optimization performance of DEPSO, and it is compared with jDE, aDE, CoDE, EPSDE, DEMPSO and IMSaDE. This section uses the same parameter setting as in Section 5.2.1. The experimental results (Mean, STD and Wilcoxon's test results) are shown in Table 10, and the results for Friedman's test and Kruskal–Wallis's test are given in Table 11. In addition, the evolution curves are plotted in Fig. 6.

Table 10
Experimental results of jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 100D functions.

Functions	Gen	Mean(STD)						
		jDE	aDE	CoDE	EPSDE	DEMPSO	IMSaDE	DEPSO
f_1	1000	1.96E−05(9.47E−06)−	5.31E−03(1.22E−03)−	2.86E+04(2.08E+03)−	2.83E−04(1.94E−04)−	3.21E−02(6.47E−02)−	6.49E−13(9.14E−13)−	2.75E−73(1.08E−72)
f_2	1000	5.48E+04(3.18E+04)−	1.19E+05(3.33E+04)−	2.34E+05(1.41E+04)−	5.08E+04(1.05E+04)−	1.67E+03(5.97E+02)−	3.41E+04(9.01E+03)−	1.62E−42(8.71E−42)
f_3	1000	6.25E−02(2.66E−02)−	1.15E+01(3.22E+00)−	6.26E+06(4.19E+05)−	9.27E−01(5.53E−01)−	4.75E+03(4.57E+03)−	1.18E−09(1.34E−09)−	3.83E−71(1.89E−70)
f_4	1000	9.38E−04(2.69E−04)−	1.63E−02(2.65E−03)−	1.24E+02(4.76E+00)−	8.66E−03(7.52E−03)−	2.84E+00(1.05E+00)−	8.67E−08(7.54E−08)−	6.57E−40(3.40E−39)
f_5	2000	4.18E+00(1.02E+00)−	1.12E+01(2.09E+00)−	6.72E+01(1.06E+00)−	1.61E+01(2.03E+00)−	1.44E+01(1.61E+00)−	2.50E+01(3.06E+00)−	2.54E−65(1.02E−64)
f_6	500	1.27E+00(1.57E+00)−	4.24E+01(6.17E+00)−	5.86E+04(2.55E+03)−	2.20E+01(2.24E+01)−	5.90E+02(1.39E+02)−	1.17E+01(8.81E+00)−	0.0E+00(0.0E+00)
f_7	2000	2.98E−02(5.54E−03)−	6.34E−02(9.96E−03)−	1.31E+01(1.21E+00)−	6.21E−02(1.20E−02)−	1.54E−01(3.72E−02)−	5.68E−02(1.42E−02)−	1.86E−03(9.15E−04)
f_8	2000	1.83E+02(5.86E+01)−	1.80E+02(6.01E+01)−	1.66E+09(2.26E+08)−	2.18E+02(6.41E+01)−	2.38E+02(1.39E+02)−	1.36E+02(4.83E+01)+	1.63E+02(5.09E+01)
f_9	1000	1.27E−05(5.75E−06)+	5.07E−03(4.56E−03)+	2.58E+02(1.59E+01)−	7.64E−03(1.14E−02)+	2.18E−02(2.36E−02)−	2.79E−03(7.59E−03)+	1.81E−02(3.31E−02)
f_{10}	1000	5.69E−04(1.09E−04)−	1.07E−02(1.44E−03)−	1.53E+01(1.56E−01)−	9.95E−01(4.38E−01)−	6.52E+00(6.63E−01)−	1.02E+00(5.76E−01)−	1.79E−11(9.64E−11)
f_{11}	1000	7.42E−08(3.16E−08)−	2.30E−05(6.95E−06)−	9.59E+01(4.70E+00)−	1.80E−06(1.63E−06)−	8.60E−05(1.18E−04)−	1.01E−14(7.82E−15)−	0.0E+00(0.0E+00)
f_{12}	500	4.06E+00(5.16E−01)−	9.24E+00(6.45E−01)−	1.08E+02(1.62E+00)−	9.53E+00(3.28E+00)−	4.73E+01(2.82E+00)−	2.95E+00(1.35E+00)−	0.0E+00(0.0E+00)
f_{13}	500	1.45E−04(5.01E−05)−	1.58E−03(3.03E−04)−	2.78E+00(1.66E−01)−	2.05E−04(6.83E−05)−	2.69E−04(1.33E−04)−	1.74E−08(1.92E−08)−	0.0E+00(0.0E+00)
f_{14}	500	2.96E+00(2.32E−01)−	5.57E+00(3.17E−01)−	2.84E+01(8.09E−01)−	3.28E+00(3.45E−01)−	5.40E+00(5.60E−01)−	2.33E+00(1.98E−01)−	1.00E−01(5.46E−04)
f_{15}	1000	4.39E−06(3.22E−06)+	3.24E−03(7.26E−03)+	7.05E+07(1.27E+07)−	2.90E−01(4.23E−01)−	2.85E+00(1.17E+00)−	2.78E−01(3.08E−01)−	1.04E−02(2.45E−02)
f_{16}	1000	3.00E−02(1.61E−01)−	2.05E−02(1.01E−02)−	2.00E+08(2.51E+07)−	8.81E+00(8.70E+00)−	1.04E+02(1.82E+01)−	8.92E−01(2.63E+00)−	1.83E−03(4.09E−03)
−		14	14	16	15	16	14	
≈		0	0	0	0	0	0	
+		2	2	0	1	0	2	

Based on Wilcoxon's test results shown in Table 10, it can be observed that DEPSO outperforms jDE, aDE, EPSDE and IMSaDE on 14, 14, 15 and 14 test problems, respectively. In contrast, CoDE and DEMPSO cannot perform better than DEPSO on any problems. This is because the compared algorithms converge very slowly and suffers from premature convergence when solving high-dimensional problems. However, the self-adaptive mutation strategy and elite archive strategy in DEPSO can not only effectively maintain the population diversity in the early stage of the evolution, but also significantly improve the convergence speed and precision in the later stage of the evolution.

In addition, the statistical results from Table 11 indicate that DEPSO is the best among the seven algorithms. It can be observed from Table 8 ($D = 30$) and 10 ($D = 100$) that the overall performance of all the algorithms will decrease as the dimension of the problem increases. However, DEPSO can still obtain better results on functions $f_1 \sim f_7$ and $f_{10} \sim f_{14}$.

In summary, the average performance of DEPSO on the 100-dimensional problems is better than jDE, aDE, CoDE, EPSDE, DEMPSO and IMSaDE.

5.2.3. Comparison with JADE, jDE, SaDE and a canonical PSO

In this experiment, DEPSO is further compared with JADE [23], jDE [20], SaDE [22] and a canonical PSO [33]. The population size is set to 100 and 200 in the case of $D = 30$ and $D = 100$, respectively. 50 independent runs are performed and the size of elite population for DEPSO is set to 30. The results for $D = 30$ and $D = 100$ are listed in Tables 12 and 13, respectively. The results of JADE, jDE, SaDE and PSO are directly taken from the literature [23].

For 30D problems, PSO performs worst and cannot obtain any best solutions and second best solutions. It may be because PSO suffers from premature convergence. Both jDE and SaDE only obtain two best solutions because they converge very slowly. The performance of JADE is better than jDE, SaDE and PSO because it introduces more population diversity in the mutation operation. In contrast, DEPSO outperforms the compared algorithms and obtains 12 best solutions and 1 s best solutions. This is because DEPSO has good diversity and faster convergence speed that stem from self-adaptive mutation strategy and elite archive strategy.

For 100D problems, DEPSO can still achieve the best performance. This is mainly because the combination of self-adaptive mutation strategy and elite archive strategy can effectively maintain the population diversity and speed up convergence. jDE, SaDE and PSO perform worse than DEPSO due to the rapidly reduced population diversity. JADE without archive obtains 0 best solutions and 5 s best solutions. JADE with archive obtains 5 best solutions and 9 s best solutions. In contrast, JADE outperforms jDE, SaDE and PSO. This could be because the adaptive control parameter and the "DE/current-to-*p*best/1" mutation strategy can increase the search diversity to some extent.

Base on the above analysis, the overall performance of DEPSO is obviously better than JADE, jDE, SaDE and PSO in terms of 30D and 100D problems

5.3. Parameter study

5.3.1. Parameter analysis in selection probability

In the selection probability shown in Eq. (14), the value of τ has an important impact on the performance of DEPSO. To determine its appropriate value, the selection probability of DE/e-rand/1 (DE) and PSO mutation strategies is experimentally studied by setting τ to $\tau \in [0.1, 4.0]$ in steps of 0.1. The selection probability of two strategies is shown in Fig. 7.

From Fig. 7(a), it can be observed that the probability of DE/e-rand/1 being selected gradually increases as τ increases, while the chance of PSO to participate in mutation operation becomes smaller. When τ is small, the selection probability of PSO strategy is closer to that of DE/e-rand/1. Although PSO strategy has more chances to participate in mutation operation, it is not beneficial to maintain the population diversity, thus leading to premature convergence.

Base on the above analysis, it is still difficult to determine the appropriate value of τ. Therefore, the evolution process is further divided into three stages: early stage, middle stage and later stage, and then the selection probability of two mutation strategies during different stages is plotted in Fig. 7(b). It can be seen from this figure that in the early stage of the evolution, the selection probability of DE/e-rand/1 is closer to 100% when $\tau \geq 1.5$. Therefore, the population diversity can be effectively maintained, and thus DEPSO is able to search more promising regions as far as possible. In the middle stage of the evolution, both strategies should be involved in mutation so that DEPSO can speed up the search while maintaining the global exploration ability. Therefore, τshould not be too large ($\tau < 2.5$). Otherwise the selection probability of PSO strategy becomes smaller, and thus it is not beneficial to converge. In the later stage of the evolution, DEPSO should focus on enhancing the local exploitation ability to improve convergence speed and precision. Therefore, the selection probability of PSO strategy should be increased and τ is set to $\tau \leq 2.2$.

In summary, a larger value of τ is beneficial to maintain the population diversity and improve the global exploration ability. A smaller value of τ is helpful to enhance the local exploitation ability and speed up the convergence.$\tau \in [1.5, 2.2]$ is recommended to balance global exploration and local exploitation.

5.3.2. The size of elite population in DEPSO

In this section, the effect of *SEP* (the size of elite population) on the performance of DEPSO is investigated by setting $SEP = \alpha \cdot NP$, $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$, thus determining the appropriate range of *SEP*. The dimension of the problem is set to $D = 30$ and the population size is set to $NP = 100$. The Mean and STD obtained by DEPSO with different values of *SEP* are given in Table 14. The MNEG and SR are listed in Table 15, and the evolution curves for test functions are plotted in Fig. 8.

(1) When $\alpha = 0.1$, DEPSO obtains 11 best solutions and 1 s best solutions. For $f_1 \sim f_5$, f_{11}, f_{13} and f_{14}, the convergence speed of DEPSO is the fastest. However, the convergence precision and reliability are poor when solving multimodal function f_{16}. Therefore, a smaller value of *SEP* ensures that DEPSO has faster convergence speed and is suitable for unimodal functions, but there may be premature convergence when the problem is multimodal.

(2) When $\alpha = 0.3$, the number of best solutions and second best solutions is second only to $\alpha = 0.1$. Although the convergence speed of DEPSO on most functions is slower than that of $\alpha = 0.1$, the higher SR is obtained when solving multimodal problems f_9 and f_{16}. Compared with $\alpha = 0.1$, the increase in the size of elite population enables the population diversity to be effectively maintained, thus improving the reliability of the algorithm.

(3) When $\alpha \in 0.5, 0.7, 0.9$, a larger size of elite population leads to the decrease of convergence precision on unimodal problems $f_1 \sim f_5$. In addition, the convergence speed on most problems is obviously slower.

Based on the above analysis, DEPSO can work best with values $\alpha \in [0.1, 0.5)$. When the problem is unimodal or low-dimensional, a smaller value of *SEP* ($SEP \in [0.1, 0.2] \cdot NP$) should be taken so that the algorithm has faster convergence speed. When the problem is multimodal or high-dimensional, a larger value of *SEP* ($SEP \in [0.4, 0.5) \cdot NP$) should be taken to enhance

Table 11
The results of Friedman and Kruskal–Wallis tests for jDE, aDE, CoDE, EPSDE, DEMPSO, IMSaDE and DEPSO on 100D functions.

Algorithms	jDE	aDE	CoDE	EPSDE	DEMPSO	IMSaDE	DEPSO
Friedman (Rank)	2.69	4.25	7.00	4.31	5.50	2.81	**1.44**
Kruskal–Wallis (Rank)	46.69	56.44	94.69	58.12	69.00	49.25	**21.31**

Table 12
Experimental results of JADE, jDE, SaDE, PSO and DEPSO on 30D functions.

Functions	Gen	Mean(STD)					
		JADE w/o archive	JADE with archive	jDE	SaDE	PSO	DEPSO
f_1	1500	*1.8E−60(8.4E−60)*	1.3E−54(9.2E−54)	2.5E−28(3.5E−28)	4.5E−20(6.9E−20)	9.6E−42(2.7E−41)	**7.60E−153(3.92E−152)**
f_2	5000	5.7E−61(2.7E−60)	*6.0E−87(1.9E−86)*	5.2E−14(1.1E−13)	9.0E−37(5.43E−36)	2.5E−19(3.9E−19)	**0.0E+00(0.0E+00)**
f_4	2000	*1.8E−25(8.8E−25)*	3.9E−22(2.7E−21)	1.5E−23(1.0E−23)	1.9E−14(1.05E−14)	9.3E−21(6.3E−20)	**3.59E−113(2.29E−112)**
f_5	5000	8.2E−24(4.0E−23)	*4.3E−66(1.2E−65)*	1.4E−15(1.0E−15)	7.4E−11(1.82E−10)	4.4E−14(9.3E−14)	**9.14E−226(0.0)**
f_6	100	*2.9E+00(1.2E+00)*	5.6E+00(1.6E+00)	1.0E+03(2.2E+02)	9.3E+02(1.8E+02)	4.5E+01(2.4E+01)	**0.0E+00(0.0E+00)**
	1500	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	8.0E−02(2.7E−01)	**0.0E+00(0.0E+00)**
f_7	3000	**6.4E−04(2.5E−04)**	*6.8E−04(2.5E−04)*	3.3E−03(8.5E−04)	4.8E−03(1.2E−03)	2.5E−03(1.4E−03)	8.97E−04(6.57E−04)
f_8	3000	**8.0E−02(5.6E−01)**	3.2E−01(1.1E+00)	1.3E+01(1.4E+01)	2.1E+01(7.8E+00)	2.5E+01(3.2E+01)	*1.23E−01(2.10E−01)*
f_9	500	**9.9E−08(6.0E−07)**	2.0E−04(1.4E−03)	*1.9E−05(5.8E−05)*	7.8E−04(1.2E−03)	1.3E−02(1.7E−02)	2.46E−03(5.09E−03)
	3000	**0.0E+00(0.0E+00)**	2.0E−04(1.4E−03)	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	1.1E−02(1.6E−02)	2.41E−03(5.13E−03)
f_{10}	500	*8.2E−10(6.9E−10)*	3.0E−09(2.2E−09)	3.5E−04(1.0E−04)	2.7E−03(5.1E−04)	4.6E−01(6.6E−01)	**4.00E−15(0.0)**
	2000	*4.4E−15(0.0E+00)*	*4.4E−15(0.0E+00)*	4.7E−15(9.6E−16)	4.3E−14(2.6E−14)	4.6E−01(6.6E−01)	**4.00E−15(0.0)**
f_{15}	500	*4.6E−17(1.9E−16)*	3.8E−16(8.3E−16)	1.6E−07(1.5E−07)	1.9E−05(9.2E−06)	1.9E−01(3.9E−01)	**1.57E−32(8.21E−48)**
	1500	*1.6E−32(5.5E−48)*	*1.6E−32(5.5E−48)*	2.6E−29(7.5E−29)	1.2E−19(2.0E−19)	1.9E−01(3.9E−01)	**1.57E−32(8.21E−48)**
f_{16}	500	*2.0E−16(6.5E−16)*	1.2E−15(2.8E−15)	1.5E−06(9.8E−07)	6.1E−05(2.0E−05)	2.9E−03(4.8E−03)	**1.35E−32(1.09E−47)**
	1500	*1.4E−32(1.1E−47)*	*1.4E−32(1.1E−47)*	1.9E−28(2.2E−28)	1.7E−19(2.4E−19)	2.9E−03(4.8E−03)	**1.35E−32(5.47E−48)**
Best		5	1	2	2	0	12
Second best		9	6	1	0	0	1

Table 13
Experimental results of JADE, jDE, SaDE, PSO and DEPSO on 100D functions.

Functions	Gen	Mean(STD)					
		JADE w/o archive	JADE with archive	jDE	SaDE	PSO	DEPSO
f_1	2000	1.2E−48(1.5E−48)	*5.4E−67(1.6E−66)*	5.0E−15(1.7E−15)	2.9E−08(3.2E−08)	6.0E−11(2.5E−10)	**6.22E−171(0.0E+00)**
f_2	8000	1.2E−26(2.0E−26)	*2.2E−37(2.5E−37)*	5.4E−02(2.7E−02)	2.4E−13(5.2E−13)	1.2E+02(6.7E+01)	**0.0E+00(0.0E+00)**
f_4	3000	1.1E−41(5.1E−41)	*9.2E−51(2.2E−50)*	4.1E−15(1.1E−15)	1.7E−05(3.8E−06)	2.8E−04(1.3E−03)	**2.38E−157(6.36E−157)**
f_5	15000	1.9E−02(1.5E−02)	*3.2E−71(8.3E−71)*	3.1E−09(5.9E−10)	1.1E+00(4.0E−01)	4.9E+01(2.5E+01)	**4.94E−324(0.0E+00)**
f_6	100	*1.1E+02(1.5E+01)*	1.2E+02(1.3E+01)	7.1E+04(6.0E+03)	3.3E+04(2.1E+03)	1.9E+04(4.5E+03)	**0.0E+00(0.0E+00)**
	1500	1.6E−01(3.7E−01)	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	4.7E+01(7.9E+01)	**0.0E+00(0.0E+00)**
f_7	6000	1.1E−03(2.1E−04)	*7.8E−04(1.4E−04)*	8.1E−03(9.0E−04)	1.0E−02(4.9E−03)	9.2E−03(2.6E−03)	**5.51E−04(2.61E−04)**
f_8	6000	5.6E−01(1.4E+00)	*4.0E−01(1.2E+00)*	7.2E+01(1.1E+01)	9.4E+01(4.0E−01)	1.3E+02(4.8E+01)	**3.93E−01(1.18E+000)**
f_9	500	*3.9E−04(2.0E−03)*	**1.5E−04(1.0E−03)**	1.1E+00(2.0E−02)	1.1E+00(1.8E−02)	1.0E+00(5.6E−01)	4.04E−03(7.86E−03)
	3000	3.9E−04(2.0E−03)	1.5E−04(1.0E−03)	**0.0E+00(0.0E+00)**	*8.6E−13(8.2E−13)*	8.8E−02(2.5E−01)	8.37E−03(1.08E−02)
f_{10}	500	7.9E−06(2.6E−06)	*4.2E−07(1.2E−07)*	8.5E−01(1.2E−01)	1.6E+00(1.2E−01)	3.6E+00(9.3E−01)	**2.74E−09(1.91E−08)**
	3000	8.9E−15(2.1E−15)	*8.0E−15(0.0E+00)*	9.9E−14(2.0E−14)	2.1E−07(1.0E−07)	2.6E+00(6.8E−01)	**5.77E−15(1.78E−15)**
f_{15}	500	*2.2E−11(1.2E−11)*	**2.8E−13(9.8E−13)**	4.0E+00(6.8E−01)	2.4E+00(3.9E−01)	1.1E+01(3.4E+00)	9.34E−03(2.00E−02)
	3000	4.7E−33(2.2E−34)	**4.7E−33(6.8E−49)**	1.7E−25(7.7E−26)	1.4E−11(9.1E−12)	1.3E−01(1.6E−01)	**4.7E−33(6.8E−49)**
f_{16}	500	*1.9E−09(1.5E−09)*	**5.8E−12(5.5E−12)**	3.1E+01(7.8E+00)	1.2E+01(1.8E+00)	9.8E+01(2.4E+01)	1.98E−03(4.22E−03)
	3000	*1.4E−32(1.1E−47)*	*1.4E−32(1.1E−47)*	2.1E−24(1.5E−24)	1.3E−11(9.6E−12)	1.4E−01(5.6E−01)	**1.35E−32(5.47E−48)**
Best		0	5	2	1	0	12
Second best		5	9	0	1	0	0

the global exploration ability. When the characteristics of the problem to be optimized are unknown, the appropriate value ($SEP \in [0.2, 0.4] \cdot NP$) should be chosen to take account of convergence speed and population diversity.

6. The application of DEPSO to arrival flights scheduling

With the rapid development of civil aviation transport, the contradiction between transport demand and airspace resource capacity is becoming increasingly prominent. As a result, the proportion of flight delay is also increasing year by year. As the main means of air traffic flow management, arrival flights scheduling can effectively decrease flight delay, reduce economic losses and improve the utilization of the runway.

Arrival flights scheduling aims to optimize the landing sequence and runway of arrival flights to minimize the delay time while satisfying the minimum safety time interval. It is a typical NP problem and is very sensitive to the parameters such as the number of flights and the number of runways. FCFS (First Come First Serve) arranges the landing sequence and runway of arrival flights according to the estimated time of arrival (ETA). This method is easy to operate and is often used as a comparison benchmark.

6.1. Mathematical model of arrival flights scheduling

Suppose that the destination airport is a parallel two-runway airport. Two consecutive flights landing on the same runway must meet the minimum safety time interval specified by ICAO

Table 14
Mean and STD obtained by DEPSO with different size of elite population.

Functions	Gen	Mean(STD)				
		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
f_1	1000	**7.34E−137(3.93E−136)**	*2.34E−102(1.26E−101)*	3.59E−87(1.91E−86)	1.49E−84(4.71E−84)	1.40E−76(7.53E−76)
f_2	1000	**7.31E−129(3.94E−128)**	*6.79E−70(3.65E−69)*	3.60E−53(1.61E−52)	1.76E−43(9.05E−43)	4.77E−50(2.56E−49)
f_3	1000	**7.10E−134(2.86E−133)**	*2.25E−99(1.21E−98)*	9.77E−86(5.22E−85)	9.97E−80(4.68E−79)	9.56E−77(2.98E−76)
f_4	1000	**3.90E−76(1.86E−75)**	*4.55E−57(1.91E−56)*	3.31E−49(1.12E−48)	1.50E−45(6.20E−45)	2.78E−45(8.84E−45)
f_5	2000	**1.04E−145(4.92E−145)**	*2.99E−88(1.37E−87)*	6.83E−69(3.68E−68)	3.72E−72(2.00E−71)	1.42E−68(5.17E−68)
f_6	500	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**
f_7	2000	2.16E−03(1.41E−03)	1.85E−03(1.29E−03)	1.63E−03(1.17E−03)	*1.49E−03(1.11E−03)*	**8.43E−04(5.14E−04)**
f_8	2000	5.80E−01(1.36E+00)	5.76E−01(1.36E+00)	**3.66E−01(1.06E+00)**	*3.75E−01(1.09E+00)*	1.41E+01(2.28E+01)
f_9	1000	9.00E−03(1.74E−02)	1.40E−03(3.16E−03)	8.21E−04(3.14E−03)	*7.40E−04(2.22E−03)*	**2.47E−04(1.33E−03)**
f_{10}	1000	**4.00E−15(0.0E+00)**	**4.00E−15(0.0E+00)**	**4.00E−15(0.0E+00)**	**4.00E−15(0.0E+00)**	*3.76E−15(8.86E−16)*
f_{11}	1000	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**
f_{12}	500	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**
f_{13}	500	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**	**0.0E+00(0.0E+00)**
f_{14}	500	9.99E−02(1.31E−04)	9.99E−02(1.93E−05)	*9.99E−02(5.31E−06)*	9.99E−02(1.19E−05)	**9.99E−02(2.24E−06)**
f_{15}	1000	**1.57E−32(8.21E−48)**	**1.57E−32(8.21E−48)**	**1.57E−32(8.21E−48)**	**1.57E−32(8.21E−48)**	**1.57E−32(8.21E−48)**
f_{16}	1000	*2.93E−03(4.86E−03)*	**1.35E−32(5.47E−48)**	**1.35E−32(5.47E−48)**	**1.35E−32(5.47E−48)**	**1.35E−32(5.47E−48)**
Best		11	7	8	7	9
Second		1	5	1	3	1

Table 15
SR and MNEG obtained by DEPSO with different size of elite population.

Functions	$\alpha = 0.1$		$\alpha = 0.3$		$\alpha = 0.5$		$\alpha = 0.7$		$\alpha = 0.9$	
	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG	SR/%	MNEG
f_1	**100**	**140**	100	198	100	267	100	350	100	435
f_2	**100**	**313**	100	403	100	616	100	666	100	669
f_3	**100**	**176**	100	248	100	339	100	434	100	500
f_4	**100**	**192**	100	272	100	376	100	473	100	532
f_5	**100**	**713**	100	770	100	999	100	1056	100	1100
f_6	100	194	**100**	**91**	100	102	100	134	100	172
f_7	100	1130	100	1155	100	1198	100	1159	**100**	**1098**
f_8	**83**	N/A	**83**	N/A	13	N/A	0	N/A	0	N/A
f_9	53	N/A	83	N/A	93	N/A	90	N/A	**97**	N/A
f_{10}	100	420	**100**	**298**	100	402	100	493	100	548
f_{11}	**100**	**113**	100	164	100	219	100	289	100	372
f_{12}	100	311	**100**	**289**	100	336	100	359	100	370
f_{13}	**100**	**92**	100	136	100	180	100	223	100	254
f_{14}	**100**	**78**	100	107	100	152	100	199	100	227
f_{15}	100	304	**100**	**214**	100	244	100	324	100	408
f_{16}	73	N/A	**100**	**210**	100	265	100	346	100	433

Table 16
Minimum safety time interval (s).

Front flight	Latter flight		
	H	M	L
H	94	114	167
M	74	74	138
L	74	74	98

(International Civil Aviation Organization), as shown in Table 16, in which H, M and L indicate that the type of the flight is heavy, medium and light, respectively.

To conveniently describe the mathematical model of arrival flights scheduling, the following variables are defined: $F = F_1, F_2, \ldots, F_N$ is a set of arrival flights. $ETA = E_1, E_2, \ldots, E_N$ is a set of the estimated time of arrival flights. $STA = S_1, S_2, \ldots, S_N$ is a set of the scheduled time of arrival flights. $T_{i,j}$ is the minimum safety time interval. Therefore, the mathematical model shown in Eq. (18) is constructed to minimize the total delay time of arrival flights.

$$\min T_{\text{delay}} = \min \sum_{i=1}^{N} (S_i - E_i)^2 \tag{18}$$

s. t.

$$S_i \geq E_i \tag{19}$$

$$S_i - E_i \leq T_{\max} \tag{20}$$

$$S_j - S_i \geq T_{i,j} \tag{21}$$

where Eq. (19) indicates that the flight F_i cannot land in advance. Eq. (20) indicates that the delay time of the flight F_i cannot exceed the specified maximum delay time $T_{\max}$. Eq. (21) indicates that the consecutive flights F_i and F_j landing on the same runway must meet the minimum safety time interval $T_{i,j}$.

The procedures to solve arrival flights scheduling using the proposed DEPSO are as follows:

Step 1: The population size NP, the dimension of the problem D, maximum number of evolution generations $G_{\max}$ and the size of elite population SEP are set. The counter for the number of evolution generations G is set to $G = 0$ and the counter for the number of individual stagnation generations NS_i is set to $NS_i = 0$.

Step 2: The initial population $\mathbf{STA}^G = \mathbf{STA}_1^G, \mathbf{STA}_2^G, \ldots, \mathbf{STA}_{NP}^G$, $\mathbf{STA}_i^G = S_1^G, S_2^G, \ldots, S_D^G$ $(i \in 1, 2, \ldots, NP)$ is randomly generated within the range $[E_i, E_i + T_{\max}]$. S_j^G, $j \in 1, 2, \ldots, D$ represents the scheduled time of flight F_j. The fitness of $\mathbf{STA}_i^G$ is calculated according to Eq. (18) and then $gbest = S_1^G, S_2^G, \ldots, S_D^G$ is used to store the individual with the minimum fitness.

Step 3: For each individual $\mathbf{STA}_i^G$, new individual $\mathbf{STA}_i^{G+1} = S_1^{G+1}, S_2^{G+1}, \ldots, S_D^{G+1}$ is generated using the Eqs. (6) and (13). If S_j^{G+1} cannot satisfy the Eqs. (19) and (20), then set $S_j^{G+1} = rand(E_j, E_j + T_{\max})$. If S_j^{G+1} cannot satisfy Eq. (21), and then set $S_j^{G+1} = S_{j-1}^{G+1} + T_{j-1,j}$.

Table 17
Optimization results of arrival flights scheduling.

ACID	Type	ETA	FCFS		DE		DEPSO	
			STA	Delay Time (s)	STA	Delay Time (s)	STA	Delay Time (s)
HBH3313	M	12: 00: 00	12: 00: 00	0	12: 00:00	0	12: 00:00	0
CHB6257	M	12 :00: 00	12: 01: 14	74	12: 01:14	74	12: 01:14	74
CSN3781	H	12: 02: 00	12: 02: 28	28	12: 02:28	28	12: 02:28	28
CHH7092	M	12: 05: 00	12:05:00	0	12:05:00	0	12:05:00	0
GCR7597	H	12: 06: 00	12:06:14	14	12: 06:14	14	12: 06:14	14
CHH7626	L	12: 08: 00	12:09:01	61	12: 09:01	61	12: 09:01	61
CSC8850	M	12: 08: 00	12:10:15	135	12: 10:15	135	12: 10:15	135
LKE9986	M	12: 09: 00	12:11:29	149	12: 11:29	149	12: 11:29	149
CSN6282	H	12: 15: 00	12:15:00	0	12: 15:00	0	12: 15:00	0
CES2136	H	12: 18: 00	12:18:00	0	12: 18:00	0	12: 19:14	74
GCR6490	M	12: 18: 00	12:19:54	114	12: 19:54	114	12: 18:00	0
CHH7676	L	12: 22: 00	12:22:12	12	12: 22:12	12	12: 22:01	1
CCA1145	M	12: 25: 00	12:25:00	0	12: 26:14	74	12: 26:14	74
CSZ9124	L	12: 25: 00	12:27:18	138	12: 25:00	0	12: 25:00	0
CES5278	H	12: 25: 00	12:28:32	212	12: 27:28	148	12: 27:28	148
AMU016	M	12: 30: 00	12:30:26	26	12: 30:00	0	12: 30:00	0
CDG4978	H	12: 31: 00	12:31:40	40	12: 33:32	152	12: 33:32	152
CSN3703	L	12: 31: 00	12:34:27	207	12: 32:18	78	12: 32:18	78
CCA4181	H	12: 32: 00	12:35:41	221	12: 35:06	186	12: 35:06	186
CES2409	M	12: 34: 00	12:37:35	215	12: 37:00	180	12: 37:00	180
CSN6540	M	12: 35: 00	12:38:49	229	12: 38:14	194	12: 38:14	194
Total delay time (s)			1875		1599		1548	

Table 18
Comparison results of FCFS, DE and DEPSO on 20 runs.

Algorithms	FCFS	DE	DEPSO
Maximum delay time (s)	1875	1599	1548
Minimum delay time (s)	1875	1548	1548
Average delay time (s)	1875	1559	1548
Friedman (Rank)	3.00	1.83	1.17

Step 4: Generate the next population $\mathbf{STA}^{G+1}$ according to the Eqs. (8) and (18). If $fitness_i^{G+1}$ is less than $fitness_i^G$, then set $NS_i = 0$ and update $gbest = \mathbf{STA}_i^{G+1} = S_1^{G+1}, S_2^{G+1}, \ldots, S_D^{G+1}$. Otherwise, $NS_i = NS_i + 1$ is set. In addition, the control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ should be updated according to the Eqs. (15) and (16).

Step 5:S et $G = G + 1$. If $G > G_{\max}$, then the algorithm is terminated and *gBest* is output. Otherwise, go to Step3.

6.2. Optimization results of arrival flights scheduling

In this experiment, 21 arrival flights between 12:00 and 12:35 in Xi'an Xianyang International Airport are used to evaluate the optimization performance of DEPSO, and it is compared with FCFS and the conventional DE. The population size is set to $NP = 80$ and the maximum number of evolution generations is set to $G_{\max} = 200$. 20 independent runs are carried out. $SEP = 30$ is set for DEPSO. The parameter settings of DE are set to $F = 0.5$ and $CR = 0.9$. Tables 17 and 18 present the optimization results and Fig. 9 illustrates the evolution curve of each algorithm. Furthermore, Friedman's test is performed in Table 18.

It can be seen from Table 18 that the total delay time of arrival flights obtained by DEPSO decreases by 17.4% compared with that of FCFS. The total delay time obtained by DEPSO decreases by 0.7% compared with DE. In addition, the maximum total delay time obtained by DEPSO on 20 independent runs is the same as the minimum total delay time, which indicates that DEPSO has better robustness. Moreover, the result of Friedman's test shows that DEPSO is significantly better than the compared algorithms. It can be observed from Fig. 9 that DEPSO has stronger global search ability and faster convergence speed. In conclusion, DEPSO can effectively decrease the total delay time of arrival flights, thus reducing the economic losses and environmental pollution caused by more fuel consumption.

7. Conclusions

The optimization performance of the conventional DE is sensitive to its mutation strategies and control parameters. To address this issue and further improve its performance, a self-adaptive mutation differential evolution algorithm based on particle swarm optimization (DEPSO) is proposed in this paper. In the early stage of the evolution, DEPSO can effectively utilize the improved DE/rand/1 mutation strategy to explore more promising regions and thus improve the ability to escape from the local optimum. In the later stage of the evolution, the PSO mutation strategy is used effectively to accelerate convergence speed and improve convergence precision. Furthermore, parameter adaptation strategy based on individual evolution status can make DEPSO more suitable for dealing with different optimization problems.

The proposed DEPSO is compared with the conventional DE, PSO, jDE, aDE, JADE, SaDE, CoDE, EPSDE, DEMPSO, IMSaDE and a canonical PSO on 30-dimensional and 100-dimensional test functions. The experimental results show that the overall performance of DEPSO is better than those of the other compared algorithms. In addition, the effect of the parameters on the performance of DEPSO is experimentally studied and the appropriate ranges of the parameters are given. The proposed DEPSO is also used to solve arrival flights scheduling problem, and the optimization results indicate that it can optimize the sequence of arrival flights and decrease the delay time, thus realizing the fuel savings.

In the future, the proposed DEPSO will be further used to solve optimization problems from air traffic management, especially the optimization of arrival/departure flights scheduling. Moreover, DEPSO can also be combined with other optimization algorithms to further improve the ability to solve different optimization problems. In addition, DE/rand/1 and DE/best/1 or DE/rand/2 and DE/best/2 could be combined to form a new adaptive mutation strategy due to the simple structure of self-adaptive mutation strategy in DEPSO, and the related research is in progress.

Acknowledgments

The authors are grateful to the reviewers for their critical and constructive review of the manuscript.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105496.

Funding

This work was supported by the National Natural Science Foundation of China (71573184).

References

[1] R. Storn, K.V. Price, Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, International Computer Science Institute Technology Report, TR-95-012, 1995.
[2] R. Storn, K.V. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[3] S. Das, A. Konar, Automatic image pixel clustering with an improved differential evolution, Appl. Soft Comput. 9 (11) (2009) 226–236.
[4] U. Maulik, I. Saha, Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery, Pattern Recognit. 42 (9) (2009) 2135–2149.
[5] A. Al-Ani, A. Alsukker, R.N. Khushaba, Feature subset selection using differential evolution and a wheel based search strategy, Swarm Evol. Comput. 9 (2013) 15–26.
[6] I. Saha, U. Maulik, S. Bandyopadhyay, D. Plewczynski, SVMeFC: SVM ensemble fuzzy clustering for satellite image segmentation, IEEE Geosci. Remote Sens. Lett. 9 (1) (2011) 52–55.
[7] S. Sarkar, S. Das, Multilevel image thresholding based on 2d histogram and maximum Tsallis entropy-a differential evolution approach, IEEE Trans. Image Process. 22 (12) (2013) 4788–4797.
[8] A. Datta, S. Ghosh, A. Ghosh, Self-adaptive differential evolution for feature selection in hyperspectral image data, Appl. Soft Comput. 13 (4) (2013) 1969–1977.
[9] G.Y. Yang, Z.Y. Dong, K.P. Wong, A modified differential evolution algorithm with fitness sharing for power system planning, IEEE Trans. Power Syst. 23 (2) (2008) 514–522.
[10] G. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inform. Sci. 329 (2016) 329–345.
[11] M.A.K. Azad, E.M.G.P. Fernandes, A modified differential evolution based solution technique for economic dispatch problems, J. Ind. Manage. Optim. 8 (4) (2012) 1017–1038.
[12] W.S. Sakr, R.A. EL-Sehiemy, A.M. Azmy, Adaptive differential evolution algorithm for efficient reactive power management, Appl. Soft Comput. 53 (2017) 336–351.
[13] T. Marcic, B. Stumberger, G. Stumberger, Differential evolution based parameter identification of a line-start IPM synchronous motor, IEEE Trans. Ind. Electron. 61 (11) (2014) 5921–5929.
[14] K.M.A. Kadhar, S. Baskar, S.M.J. Amali, Diversity controlled self-adaptive differential evolution based design of non-fragile multivariable PI controller, Eng. Appl. Artif. Intell. 46 (2015) 209–222.
[15] X.G. Zhou, G.J. Zhang, X.H. Hao, L. Yu, D.W. Xu, Differential evolution with multi-stage strategies for global optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, Vancouver, BC, 2016, pp. 2550–2557.
[16] M. Ali, P. Siarry, M. Pant, An efficient differential evolution based algorithm for solving multi-objective optimization problems, European J. Oper. Res. 217 (2) (2012) 404–416.
[17] R. Gamperle, S.D. Muller, P. Koumoutsakos, A parameter study for differential evolution, in: Proceedings of WSEAS International Conference on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS, New York, 2002, pp. 293–298.
[18] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[19] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. 9 (6) (2005) 448–462.
[20] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[21] N. Nasimul, B. Danushka, I. Hitoshi, An adaptive differential evolution algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, New Orleans, LA, 2011, pp. 2229–2236.
[22] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, Edinburgh, Scotland, UK, 2005, pp. 1785–1791.
[23] J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[24] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.
[25] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[26] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with discrete mutation control parameters, Expert Syst. Appl. 42 (3) (2015) 1551–1572.
[27] S. Wang, Y. Li, H. Yang, H. Liu, Self-adaptive differential evolution algorithm with improved mutation strategy, Soft Comput. 22 (10) (2018) 3433–3447.
[28] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[29] H. Liu, Z.X. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. 10 (2010) 629–640.
[30] G. Lin, J. Zhang, Z. Liu, Hybrid particle swarm optimization with differential evolution for numerical and engineering optimization, Int. J. Autom. Comput. 15 (1) (2016) 103–114.
[31] B. Mao, Z. Xie, Y. Wang, H. Handroos, H. Wu, S. Shi, A hybrid differential evolution and particle swarm optimization algorithm for numerical kinematics solution of remote maintenance manipulators, Fusion Eng. Des. 124 (2017) 587–590.
[32] H. Wang, L.L. Zuo, J. Liu, W.J. Yi, B. Niu, Ensemble particle swarm optimization and differential evolution with alternative mutation method, Nat. Comput. (2018), http://dx.doi.org/10.1007/s11047-018-9712-z.
[33] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inform. Process. Lett. 85 (6) (2003) 317–325.
[34] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE International Conference on Evolutionary Computation Proceedings, IEEE, Anchorage, AK, 1998, pp. 69–73.
[35] S. Rahnamayan, H.R. Tizhoosh, M. Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (1) (2008) 64–79.
[36] M. Ali, M. Pant, A. Abraham, Simplex differential evolution, Acta Polytechn. Hungar. 6 (5) (2009) 95–115.
[37] W. Gong, Z. Cai, L. Jiang, Enhancing the performance of differential evolution using orthogonal design method, Appl. Math. Comput. 206 (1) (2008) 56–69.
[38] A.B. Ozer, CIDE: Chaotically initialized differential evolution, Expert Syst. Appl. 37 (6) (2010) 4632–4641.
[39] M. Ali, M. Pant, A. Abraham, Unconventional initialization methods for differential evolution, Appl. Math. Comput. 219 (9) (2013) 4474–4494.
[40] I. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Inform. Sci. 297 (2015) 216–235.
[41] M. Asafuddoula, T. Ray, R. Sarker, An adaptive hybrid differential evolution algorithm for single objective optimization, Appl. Math. Comput. 231 (2014) 601–618.
[42] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Comput. 10 (8) (2006) 673–686.
[43] J. Brest, M.S. Maucec, Population size reduction for the differential evolution algorithm, Appl. Intell. 29 (3) (2008) 228–247.
[44] W. Zhu, Y. Tang, J.A. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Inform. Sci. 223 (2013) 164–191.
[45] A.P. Piotrowski, Review of differential evolution population size, Swarm Evol. Comput. 32 (2017) 1–24.
[46] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy, IEEE Trans. Syst. Man Cybern. C 42 (5) (2012) 744–767.
[47] J. Sun, Q. Zhang, E.P.K. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Inform. Sci. 169 (3–4) (2005) 249–262.
[48] L. Wang, Y. Xu, L. Li, Parameter identification of chaotic systems by hybrid Nelder–Mead simplex search and differential evolution algorithm, Expert Syst. Appl. 38 (4) (2011) 3238–3245.
[49] A. Ponsich, C.A.C. Coello, A hybrid differential evolution-Tabu search algorithm for the solution of job-shop scheduling problems, Appl. Soft Comput. 13 (1) (2013) 462–474.
[50] H. Guo, Y. Li, J. Li, H. Sun, D. Wang, X. Chen, Differential evolution improved with self-adaptive control parameters based on simulated annealing, Swarm Evol. Comput. 19 (2014) 52–67.
[51] M. Pant, R. Thangaraj, A. Abraham, DE-PSO: A new hybrid meta-heuristic for solving global optimization problems, New Math. Nat. Comput. 7 (3) (2011) 363–381.
[52] S. Sayah, A. Hamouda, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, Appl. Soft Comput. 13 (4) (2013) 1608–1619.

[53] G. Tian, Y. Ren, M. Zhou, Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm, IEEE Trans. Intell. Transp. Syst. 17 (11) (2016) 3009–3021.
[54] W.J. Zhang, X.F. Xie, DEPSO: hybrid particle swarm with differential evolution operator, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, IEEE, Washington, DC, 2003, pp. 3816–3821.
[55] J. Kennedy, Bare bones particle swarms, in: Proceedings of IEEE Swarm Intelligence Symposium, IEEE, Washington, DC, 2003, pp. 80–87.
[56] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.