SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •

May 2010 Vol. 53 No. 5: 980–989
doi: 10.1007/s11432-010-0114-9

An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization

XIN Bin[1,2*], CHEN Jie[1,2*], PENG ZhiHong[1,2] & PAN Feng[1,2]

[1]*School of Automation, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*Key Laboratory of Complex System Intelligent Control and Decision, Ministry of Education, Beijing* 100081*, China*

Received August 20, 2009; accepted December 4, 2009

Abstract This paper presents extensive experiments on a hybrid optimization algorithm (DEPSO) we recently developed by combining the advantages of two powerful population-based metaheuristics—differential evolution (DE) and particle swarm optimization (PSO). The hybrid optimizer achieves on-the-fly adaptation of evolution methods for individuals in a statistical learning way. Two primary parameters for the novel algorithm including its learning period and population size are empirically analyzed. The dynamics of the hybrid optimizer is revealed by tracking and analyzing the relative success ratio of PSO versus DE in the optimization of several typical problems. The comparison between the proposed DEPSO and its competitors involved in our previous research is enriched by using multiple rotated functions. Benchmark tests involving scalability test validate that the DEPSO is competent for the global optimization of numerical functions due to its high optimization quality and wide applicability.

Keywords global optimization, statistical learning, differential evolution, particle swarm optimization, hybridization, adaptation, rotated function

Citation Xin B, Chen J, Peng Z H, et al. An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Sci China Inf Sci, 2010, 53: 980–989, doi: 10.1007/s11432-010-0114-9

1 Introduction

The global optimization of numerical functions is one of the most important generic topics in scientific and engineering researches since many problems can be depicted in its forms. In a recent paper published on the journal *Sci China Ser F-Inf Sci* [1], we proposed a novel hybrid optimizer for the global optimization of numerical functions. The optimizer was designed with the purpose of combining the advantages of two powerful population-based metaheuristic algorithms—differential evolution (DE) and particle swarm optimization (PSO) [2, 3]. Our preliminary research shows that the hybrid optimizer is effective and competent for numerical optimization. The goal of this paper is to provide a comprehensive validation of the proposed algorithm and make a detailed analysis of its parameter setting and evolution dynamics. The rest of this paper is organized as follows. In section 2, a brief introduction on the PSO and DE optimizers for hybridization is provided for ease of comprehension of the proposed DEPSO algorithm.

*Corresponding authors (email: brucebin@bit.edu.cn, chenjie@bit.edu.cn)

In section 3, the proposed hybrid algorithm is presented. Its parameter setting and evolution dynamics are analyzed in the optimization of several representative functions. In section 4, numerical experiments based on a test suite, composed of 10 scalable benchmark problems of different features, are implemented to compare the performances of different optimizers. Section 5 concludes the paper.

2 A brief introduction on PSO and DE

2.1 Particle swarm optimization

The particle swarm optimizer used here for the hybridization is a variant of PSO with constriction factor (PSO-cf) whose iteration equations are shown as follows [2]:

$$v_i^d(k+1) = \chi * [v_i^d(k) + c_1 * rand1_i^d * (pbest_i^d(k) - x_i^d(k)) + c_2 * rand2_i^d * (nbest_i^d(k) - x_i^d(k))], \quad (1)$$

$$x_i^d(k+1) = x_i^d(k) + v_i^d(k+1), \quad (2)$$

where the subscript i is the indication of the ith particle ($i = 1, 2, \ldots, PS$; PS is population size); the superscript d is the indication of the dth dimension; the index k is the indication of the kth generation; $\boldsymbol{x}_i = (x_i^1, x_i^2, \ldots, x_i^D)$ and $\boldsymbol{v}_i = (v_i^1, v_i^2, \ldots, v_i^D)$ are the position and velocity of the ith particle, respectively; χ is the so-called constriction factor proposed by Clerc and Kennedy [2]; c_1 and c_2 are acceleration coefficients; $rand1_i^d$ and $rand2_i^d$ are random numbers uniformly distributed in the interval [0,1]. $\boldsymbol{pbest}_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^D)$ is the best position found so far by the ith particle; $\boldsymbol{nbest}_i = (nbest_i^1, nbest_i^2, \ldots, nbest_i^D)$ represents the best position found by the particles in the neighborhood of the ith particle. Different neighborhood topologies correspond to different PSO versions [4]. The topology applied here is the well-known von Neumann topology suggested by Kennedy and Mendes [4] in which each particle has four neighbors on a two-dimensional lattice (left, right, above and below).

Improvement strategies which give birth to more advanced PSO variants can be largely categorized into the following five types:

1) parameter tuning (e.g., time-varying inertia weight and acceleration coefficients [5] and dynamic population sizing and management [6]);

2) topology tuning (e.g., employing well-balanced *Lbest* topologies [4] and dynamic topologies [7]);

3) tuning of learning patterns (e.g., the comprehensive learning strategy in [8] and the fully informing strategy in [9]);

4) multiple-swarm strategies [10, 11];

5) hybridization strategies [12, 13].

Note that some complicated PSO variants may adopt multiple strategies above simultaneously. The improvement strategy adopted in this paper belongs to the final type—hybridization. A recent standard PSO algorithm for performance comparisons acknowledged by PSO community is the SPSO2007 available on the Particle Swarm Central: http://www.particleswarm.info.

2.2 Differential evolution

The differential evolution proposed by Storn and Price [3] is also a formidable population-based optimizer. It has three operators—mutation, crossover and selection. The mutation in DE is a distinct innovation. It is based on the difference of different individuals (solutions). A general notation for DE is DE/x/y/z where x specifies the base vector to be mutated, y is the number of difference vectors used, and z denotes the crossover scheme [3]. The most classical DE variant is DE/$rand$/1/bin. In this DE variant, for the mutation of the ith individual in the DE population $\{\boldsymbol{x}_i | i = 1, 2, \ldots, PS\}$, three different individuals $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$ and $\boldsymbol{x}_{r3}$ with $r1 \neq r2 \neq r3 \neq i$ will be randomly ($rand$) chosen from the population to generate a new vector. The new vector can be expressed as follows:

$$\boldsymbol{z}_i = \boldsymbol{x}_{r1} + F * (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}), \quad (3)$$

Table 1 Pseudocode for proposed DEPSO[a),b)]

Generate a uniformly distributed random population $P = \{x_i \mid i = 1, 2, \ldots, PS\}$ within the whole search range;
Set $Pr = 0.5$ and the generation record $k = 0$; Set the records N_s^{PSO}, N_f^{PSO}, N_s^{DE} and N_f^{DE} at zero;
Do $k = k + 1$
  For $i = 1$ **to** PS
    If $rand < Pr$ Evolve the ith individual (particle) with PSO:
      Change x_i according to (1) and (2); if certain components of $\boldsymbol{x}_i$ exceed search bound, set them to the bound and force corresponding velocity components to be zero; evaluate $f(\boldsymbol{x}_i)$; if $f(\boldsymbol{x}_i)$ is better than $f(\boldsymbol{pbest}_i)$, update $\boldsymbol{pbest}_i$ with $\boldsymbol{x}_i$; if $f(\boldsymbol{x}_i)$ is better than the current discovered best objective value, let $N_s^{PSO} = N_s^{PSO} + 1$, otherwise let $N_f^{PSO} = N_f^{PSO} + 1$; and update the ***nbest*** of all neighbors of this individual if the fitness of ***nbest*** is improved;// $f(\cdot)$ is the objective function.
    Else Evolve the ith individual with DE:
      Apply differential mutation within the scope of the personally best positions of all particles, and crossover to $\boldsymbol{pbest}_i$ according to (3) and (4), to get a trial vector $\boldsymbol{u}_i$; if certain components of $\boldsymbol{u}_i$ exceed search bound, set them to the bound; evaluate $f(\boldsymbol{u}_i)$; if $f(\boldsymbol{u}_i)$ is better than $f(\boldsymbol{pbest}_i)$, update $\boldsymbol{pbest}_i$ with $\boldsymbol{u}_i$; if $f(\boldsymbol{u}_i)$ is better than the current discovered best objective value, let $N_s^{DE} = N_s^{DE} + 1$, otherwise let $N_f^{DE} = N_f^{DE} + 1$; and update the ***nbest*** of all neighbors of this individual if the fitness of ***nbest*** is improved;
    End if
    If a learning period is over, make a statistical analysis as follows:
      If $N_s^{PSO} + N_s^{DE} == 0$ // This means neither PSO nor DE makes contribution to fitness improvement.
        Set Pr at 0.5; // A fair chance for PSO and DE to be employed
      Else
        If $N_s^{DE} + N_f^{DE} == 0$ Set Pr=1; **End**
        If $N_s^{PSO} + N_f^{PSO} == 0$ Set Pr=0; **End**
        If $N_s^{DE}(N_s^{PSO} + N_f^{PSO}) + N_s^{PSO}(N_s^{DE} + N_f^{DE}) \neq 0$
          Compute the relative success ratio (Pr) of PSO versus DE according to (5);
        End
      End if
      Set the records N_s^{PSO}, N_f^{PSO}, N_s^{DE} and N_f^{DE} at zero;
    End if
  End for
While termination criteria are not satisfied

a) Instructions for symbols used in this table can be found in sections 2 and 3;

b) The procedure presented here is slightly different from that in our previous research. The learning period is now related to the accumulated number of function evaluations instead of the number of generations. The subroutine of statistical learning is placed into the inner loop of the whole procedure. This modification provides more opportunities for the adaptation of evolution methods.

where F is the so-called scaling factor which is a positive constant usually chosen from the interval (0,1). After mutation, a binominal (bin) crossover operates on the vector $\boldsymbol{z}_i$ and the individual $\boldsymbol{x}_i$ to generate the final vector $\boldsymbol{u}_i$ in the following way:

$$u_i^d = \begin{cases} z_i^d, & \text{if } rand_i^d \leqslant CR \text{ or } d == rn_i; \\ x_i^d, & \text{otherwise,} \end{cases} \tag{4}$$

where x_i^d, z_i^d and u_i^d are the dth dimensional components of the vectors $\boldsymbol{x}_i$, $\boldsymbol{z}_i$ and $\boldsymbol{u}_i$, respectively; CR is the predefined crossover probability; rn_i is a number randomly selected from the index set $\{1, 2, \ldots, D\}$ and used to ensure that the trial vector $\boldsymbol{u}_i$ is different from the original solution $\boldsymbol{x}_i$. Finally, $\boldsymbol{u}_i$ will be compared with $\boldsymbol{x}_i$, and the better one will be selected as a member of the DE population for the next generation.

The categories of improvement strategies for DE are quite similar to those for PSO. Readers interested may refer to [3] and [14] for advanced DE variants.

3 A novel DEPSO

3.1 The idea of statistical learning

The DEPSO we proposed here adopts a statistical learning strategy which selects the evolution method for each individual according to the relative success ratio of alternative methods in a previous learning period. If the discovered best objective value is improved by PSO, the success number of PSO is increased. Otherwise, the failure number of PSO is increased. This rule is same for DE.

Denote by N_s^{PSO} and N_f^{PSO}, respectively, the success number and failure number of PSO in a learning period, and by N_s^{DE} and N_f^{DE} the DE counterparts. The relative success ratio of PSO versus DE, that is, the probability of choosing PSO as the evolution method for an individual, can be expressed as follows:

$$\Pr = \frac{N_s^{PSO}(N_s^{DE} + N_f^{DE})}{N_s^{PSO}(N_s^{DE} + N_f^{DE}) + N_s^{DE}(N_s^{PSO} + N_f^{PSO})}. \tag{5}$$

The performance of PSO and DE may vary with search stages. They may fit respectively diverse requirements on the tradeoff between exploration and exploitation at different search stages [15]. Our motive is to use the fitter method to implement the search of problem space in different stages. The statistical learning here is de facto an adaptation of search methods by feedback from the current state of search [16].

The pseudocode for the DEPSO is presented in Table 1. One of the most important parameters for the proposed DEPSO is the length of its learning period (denoted by L_P). When the accumulated number of function evaluations is divided exactly by L_P, which indicates the end of a learning period, the performance of PSO and DE during the last learning period including L_P function evaluations will be statistically analyzed. The relative success ratio of PSO versus DE obtained from the statistical learning will be utilized to guide the adaptation of evolution methods in subsequent learning period. Another important parameter for the DEPSO is the population size, denoted by PS, which is very common for all population-based metaheuristics such as PSO and DE. In the following, we will make an integrated analysis of the two key parameters by comparing the performance of the DEPSO in the optimization of several representative functions.

3.2 Parameter setting: L_P and PS

In order to achieve a rational and effective parameter setting for the proposed DEPSO, we made a preliminary test to analyze the effects of the two key parameters L_P and PS on the performance of the optimizer. Taking into account the possible coupling effect of the two parameters, we chose different combinations for their settings. The setting of population size includes $PS = 10, 30, 50, 100, 300$ and 1000. For the length of learning period, we considered $L_P = 50, 100, 300$ and 1000. In addition, a very small L_P like $L_P = 10$ is usually insufficent to ensure a convincing and reliable statistical result, so it is not involved in our comparative analysis. Therefore, there are totally $6 \times 4 = 24$ different combinations of the two parameters compared in the following experiment. We used four representative functions of different features as a test suite. The expressions for these functions are presented as follows:

1) Sphere function: $f_1(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$, $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]$ and $\boldsymbol{x} \in [-100, 100]^D$.
2) Rosenbrock function: $f_2(\boldsymbol{x}) = \sum_{i=1}^{D-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \boldsymbol{x} \in [-30, 30]^D$.
3) Rastrigin function: $f_3(\boldsymbol{x}) = \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i) + 10], \boldsymbol{x} \in [-5.12, 5.12]^D$.
4) Schwefel function: $f_4(\boldsymbol{x}) = 418.98289 * D - \sum_{i=1}^{D} x_i * \sin(\sqrt{|x_i|}), \boldsymbol{x} \in [-500, 500]^D$.

The minimal values of the above functions are all zero. For illustration, the landscapes of two-dimensional versions of the four functions are depicted in Figure 1. Note that the dimensional increase of these scalable functions does not change their basic features. The Sphere function is a simple unimodal function. The Rosenbrock function is a unimodal-like function whose landscape contains a banana-shaped valley around its optimum [8]. The Rastrigin function is a multimodal function with a rugged landscape and large numbers of local optima. Note that its local optima are regularly distributed on a unimodal bottom envelope and it is categorized as a single-funnel function in [17]. The Schwefel function is also

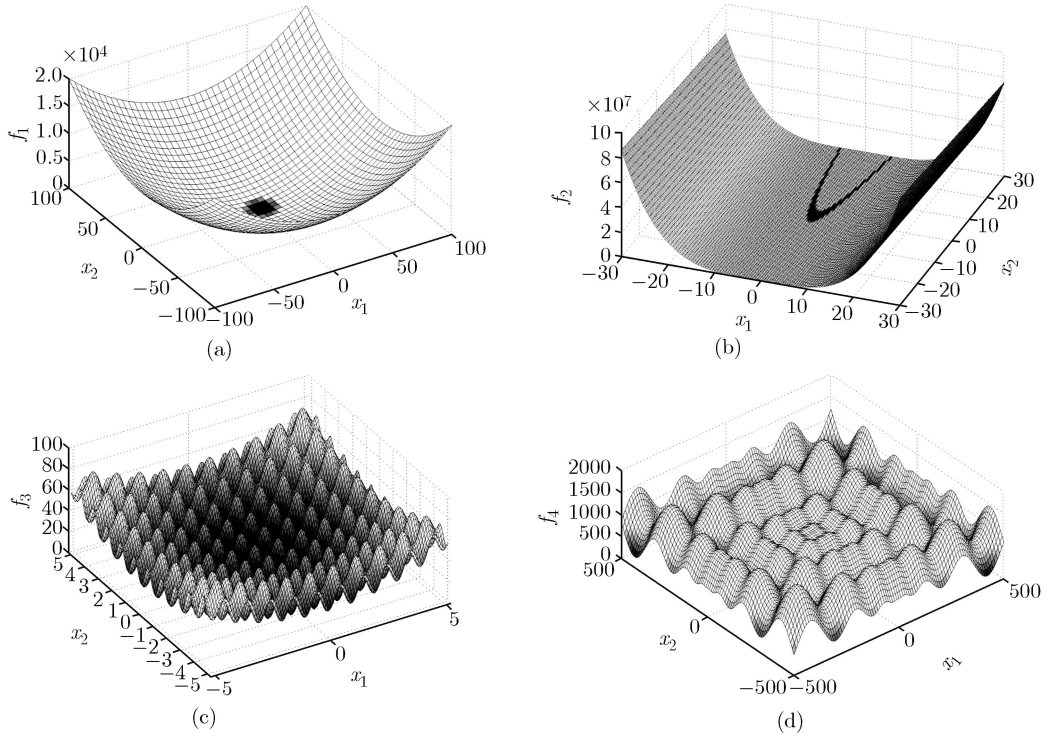

Figure 1 An illustration for 2-dimensional landscapes of four test functions. (a) 2D Sphere function; (b) 2D Rosenbrock function; (c) 2D Rastrigin function; (d) 2D Schwefel function.

a multimodal function but it has a multi-funnel landscape [17]. There are many highly attractive local optima far apart from its global optimum in its landscape. Multi-funnel functions are usually regarded to be much harder to optimize than single-funnel functions.

In this test, we chose three different dimensions, including $D = 2$, $D = 10$ and $D = 30$, for these test functions to take account of the effect of problem dimension on population size. In each case, the optimizer will be terminated when the maximal number of function evaluations (denoted by $NFE_{\max}$) is reached. It was observed in the experiement that the DEPSO was more sensitive to PS than L_P. In most cases, $PS = 30$ is a good choice for the optimizer's population sizing. The DEPSO with $PS = 30$ performs obviously better than its counterparts with other population sizes. However, a smaller PS benefits the DEPSO in the optimization of the simple unimodal Sphere function. This is because a smaller population size favors fast convergence on unimodal landscapes. In fact, a better strategy for the optimization of unimodal functions is to adopt gradient descent methods which can make best of acquired information. Here, we mainly care about the performance of optimizers in the optimization of complicated multimodal functions.

In the optimization of 10D and 30D multi-funnel Schwefel functions, the best results correspond to $PS = 50$ and $PS = 100$, respectively. Larger population sizes such as 300 and 1000 did not produce satisfactory results in all cases. Given limited iteration numbers (i.e., time resource), overlarge population sizes often result in slow convergence since each individual cannot be sufficiently evolved. In essence, the selection of population size depends on the problem-dependent tradeoff between exploration and exploitation [15]. In addition, it was also observed that the change of L_P has no obvious effect on the DEPSO's performance if its population size is properly chosen. The setting $L_P = 100$ has slight advantages over other settings ($L_P = 50, 300$ and 1000), and therefore we set L_P at 100 in subsequent comparative experiments.

The online change of the relative success ratio of PSO versus DE in the optimization of 2D test functions ($f_1 \sim f_4$) is depicted in Figure 2. From the four curves of success ratio shown in Figure 2, it is easy to see that the PSO method usually fits the evolution of individuals at early stages. In contrast, DE often has more contribution to the improvement of discovered best objective values at later stages. The ada-

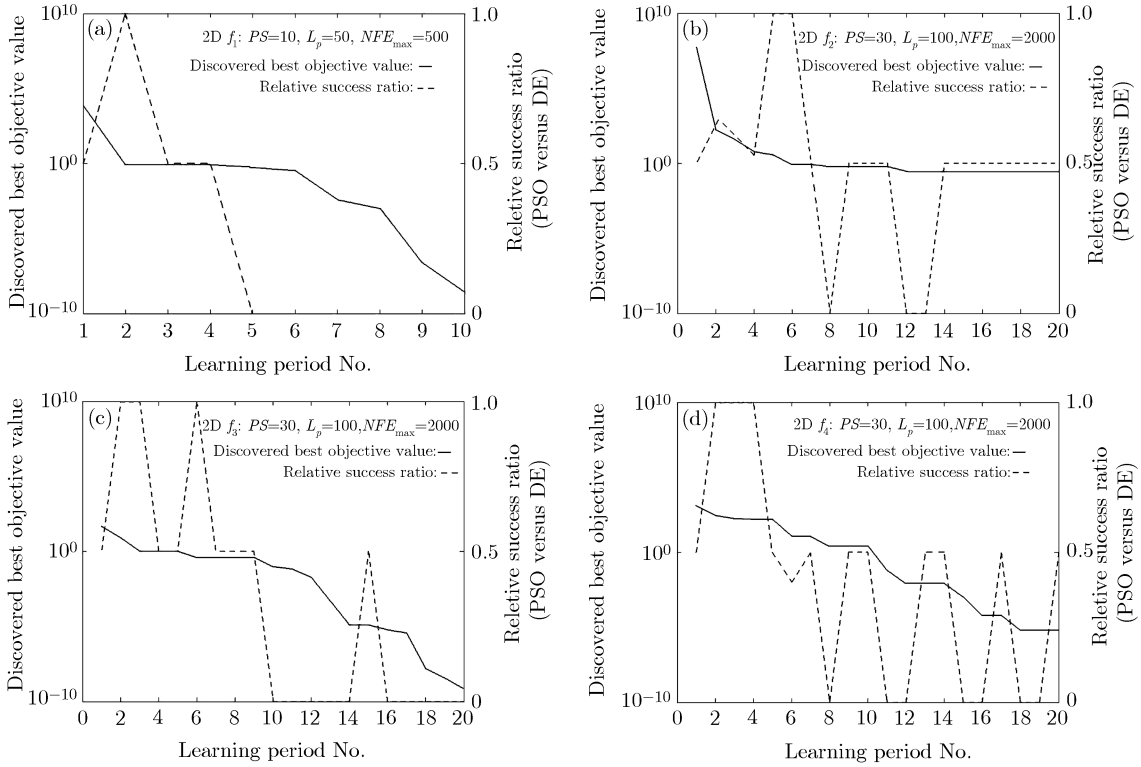

Figure 2 Dynamics of the proposed DEPSO. (a) 2D Sphere function; (b) 2D Rosenbrock function; (c) 2D Rastrigin function; (d) 2D Schwefel function.

ptive selection of evolution methods increases the diversity of evolution and brings more opportunities to discover better solutions in search space.

4 Numerical experiments

Ten representative multimodal functions are employed to show the global optimization performance of proposed DEPSO. Note that these test functions are often used as benchmark test suite in numerical experiments [2–12, 17–20]. Their expressions are presented as follows.

1) Acley function:

$$F_1(\boldsymbol{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right),\ \boldsymbol{x} \in [-32, 32]^D.$$

2) Rotated & Shifted Acley function:

$$F_2(\boldsymbol{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right),$$
$$\boldsymbol{x} \in [-32, 32]^D,\ \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}_2) * \boldsymbol{M}_2,$$

where $\boldsymbol{o}_2$ is a randomly generated shift vector located in $[-32, 32]^D$ and $\boldsymbol{M}_2$ is a $D \times D$ dimensional rotation matrix. Note that the procedure of generating rotation matrices can be found in [18].

3) Alpine function:

$$F_3(\boldsymbol{x}) = \sum_{i=1}^{D} |x_i * \sin(x_i) + 0.1 * x_i|,\ \boldsymbol{x} \in [-10, 10]^D.$$

4) Rotated & Shifted Alpine function:

$$F_4(\boldsymbol{x}) = \sum_{i=1}^{D} |z_i * \sin(z_i) + 0.1 * z_i|,\ \boldsymbol{x} \in [-10, 10]^D,\ \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}_4) * \boldsymbol{M}_4,$$

where $\boldsymbol{o}_4$ is a randomly generated shift vector located in $[-10, 10]^D$ and $\boldsymbol{M}_4$ is a $D \times D$ dimensional rotation matrix.

5) Rastrigin function (Denoted here by F_5; see subsection 3.2 for its expression).

6) Rotated & Shifted Rastrigin function:

$$F_6(\boldsymbol{x}) = \sum_{i=1}^{D} [z_i^2 - 10\cos(2\pi z_i) + 10],\ \boldsymbol{x} \in [-5.12, 5.12]^D,\ \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}_6) * \boldsymbol{M}_6,$$

where $\boldsymbol{o}_6$ is a randomly generated shift vector located in $[-5.12, 5.12]^D$ and $\boldsymbol{M}_6$ is a $D \times D$ dimensional rotation matrix.

7) Schwefel function (Denoted here by F_7; see subsection 3.2 for its expression).

8) Rotated Schwefel function:

$$F_8(\boldsymbol{x}) = 418.98289 * D + \sum_{i=1}^{D} y_i,\ \boldsymbol{x} \in [-500, 500]^D,\ \boldsymbol{z} = \boldsymbol{x} * \boldsymbol{M}_8,$$

$$y_i = \begin{cases} -z_i \sin(|z_i|^{0.5}), & \text{if } |z_i| \leqslant 500; \\ 0.001(|z_i| - 500)^2, & \text{if } |z_i| > 500, \end{cases}$$

where $\boldsymbol{M}_8$ is a $D * D$ dimensional rotation matrix.

9) Shifted Rosenbrock function:

$$F_9(\boldsymbol{x}) = \sum_{i=1}^{D-1} [100(z_{i+1} - z_i^2)^2 + (1 - z_i)^2],\ \boldsymbol{x} \in [-30, 30]^D,\ \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}_9 + \boldsymbol{i},\ \boldsymbol{i} = [1, 1, \ldots, 1]_{1\times D},$$

where $\boldsymbol{o}_9$ is a randomly generated shift vector located in $[-30, 30]^D$.

10) Shifted Levy function:

$$F_{10}(\boldsymbol{x}) = \sum_{i=1}^{D-1} (z_i - 1)^2[1 + \sin^2(3\pi z_{i+1})] + \sin^2(3\pi z_1) + \mid z_D - 1 \mid (1 + \sin^2(2\pi z_D)),$$

$$\boldsymbol{x} \in [-10, 10]^D,\ \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}_{10} + \boldsymbol{i},\ \boldsymbol{i} = [1, 1, \ldots, 1]^D,$$

where $\boldsymbol{o}_{10}$ is a randomly generated shift vector located in $[-10, 10]^D$.

All the above functions except for the Shifted Rosenbrock function contain large numbers of local optima in their landscapes. The optimal objective values of these functions for minimization are all zero. The purpose of employing rotated or/and shifted functions to analyze the performance of optimizers is to eliminate the "irrational" advantages of some special but useless approaches like those which favor convergence toward zero [19]. It is convenient to identify the irrationality of these approaches by using both unrotated unshifted functions and their rotated or/and shifted counterparts as test benchmark.

In this comparative experiment, two settings $D = 10$ and $D = 30$ are adopted for the dimension of all functions above to test the scalability of optimizers. The algorithms for comparison are listed together with settings of their parameters except for population size as follows:

1) DE/*rand*/1/*bin* ($F = 0.5$, $CR = 0.9$ [3]);

2) SPSO2007 (http://www.particleswarm.info);

3) DEPSO-ZX proposed by Zhang and Xie (parameter setting follows that in [12], e.g., $CR = 0.1$);

4) Our DEPSO ($L_P = 100$).

According to the preliminary experiment presented in subsection 3.2, two settings $PS = 30$ and $PS = 50$ are considered for the population size of our DEPSO in the case $D = 10$, and $PS = 30, 100$ for

$D = 30$. These settings are very common and also frequently used in the literature for other optimizers involved in this experiment [2, 3, 5–12, 20]. So the population sizing scheme applied for our DEPSO is also used for other optimizers.

The allowable maximal numbers of function evaluations ($NFE_{\max}$) for the optimization of all 10D and 30D functions except for Schwefel function and its rotated counterpart are set at 2×10^4 and 1×10^5, respectively. $NFE_{\max}$ is set at 5×10^4 and 3×10^5 for 10D and 30D Schwefel functions respectively, which is same for their rotated counterparts. These settings are beneficial to observing the performance differences of the optimizers involved. All optimizers will run 30 times independently and stop when $NFE_{\max}$ is reached in each run. The experimental results in the form of the mean plus standard deviation of finally discovered best objective values are presented in Tables 2 and 3, and only the results corresponding to better population sizes are reported. The figures in square brackets following the optimization results are the statistical results of t-tests for paired comparisons. If an optimizer outperforms another one with 95% confidence, the t-test result of the paired comparison is 1; the result is -1 if it is outdone with 95% confidence by another one, and 0 otherwise. The mean and standard deviation of time cost in each case are also shown in Tables 2 and 3. Note that all optimizers involved in the experiment were implemented on a PC with 2.80 GHz Pentium(R)-IV CPU and 512 MB internal memory.

As the statistical results in Tables 2 and 3 indicate, the proposed DEPSO has obvious advantages over its three competitors. In the optimization of 10D Apline function (F_3) and 10D Rotated Schwefel function (F_8), our DEPSO is inferior to DE/*rand*/1/*bin* but outperforms SPSO2007 and DEPSO-ZX.

Table 2 Statistical results of employing eight optimizers to minimize 10D benchmark functions[a)]

Optimizers	F_1	F_2	F_3	F_4	F_5
DE/*rand*/1/*bin*	(1.9e−09)± (2.5e−10)	(2.3e−09)± (3.5e−10)	**(8.3e−13)± (7.1e−13)***	**(9.3e−04)± (7.2e−04)**	(5.6e+00)± (1.1e+00)
Runtime (s)	1.6±0.0	2.6±0.0	1.4±0.0	2.3±0.0	1.6±0.0
SPSO2007	**(1.9e−13)± (2.4e−14)**	(9.3e−08)± (1.1e−08)	(6.0e−07)± (4.3e−07)	(3.9e−03)± (9.2e−04)	(1.3e+01)± (1.3e+00)
Runtime (s)	1.7±0.0	2.7±0.0	1.5±0.0	2.5±0.0	1.7±0.0
DEPSO-ZX	(4.6e−10)± (1.2e−10)	**(7.0e−10)± (1.5e−10)**	(2.5e−07)± (1.9e−07)	(4.0e−01)± (1.2e−01)	(3.3e−03)± (1.1e−03)
Runtime (s)	1.4±0.0	2.4±0.0	1.1±0.0	2.1±0.0	1.3±0.0
Our DEPSO	**(1.3e−14)± (3.1e−15)**[4]*	**(1.7e−10)± (3.6e−11)**[4]*	**(6.3e−10)± (3.0e−10)**[2]	**(5.2e−04)± (4.6e−04)**[2]*	**0±0**[4]*
Runtime (s)	1.6±0.0	2.6±0.0	1.4±0.0	2.4±0.0	1.6±0.0
	F_6	F_7	F_8	F_9	F_{10}
DE/*rand*/1/*bin*	(1.4e+01)± (1.8e+00)	(2.1e+02)± (3.2e+01)	**(2.7e−05)± (1.3e−12)***	(3.5e+00)± (2.0e−01)	(4.1e−17)± (1.3e−17)
Runtime (s)	2.5±0.1	3.9±0.0	7.2±0.1	2.0±0.0	2.0±0.0
SPSO2007	(1.9e+01)± (1.3e−17)	(4.0e+02)± (4.9e+01)	(5.0e+01)± (1.3e+01)	(4.9e+00)± (1.4e−01)	**(8.9e−25)± (3.3e−26)**
Runtime (s)	2.6±0.1	4.6±0.1	7.7±0.1	2.1±0.0	2.1±0.0
DEPSO-ZX	(1.4e+01)± (1.1e+00)	(2.4e+02)± (3.7e+01)	(1.9e+02)± (2.7e+01)	(4.8e+00)± (5.3e−01)	(1.6e−18)± (7.2e−19)
Runtime (s)	2.2±0.0	3.1±0.0	6.3±0.1	1.7±0.0	1.8±0.0
Our DEPSO	(1.4e+01)± (1.1e+00)[1]	**(2.7e−05)± (1.4e−16)**[4]*	**(1.7e−04)± (1.6e−04)**[2]	**(1.8e+00)± (2.7e−01)**[4]*	**(4.3e−27)± (7.5e−28)**[4]*
Runtime (s)	2.5±0.0	3.9±0.1	7.1±0.1	1.9±0.0	2.0±0.0

a) In each case, obviously better results are highlighted in bold and the best one is marked by asterisks. This denotation manner is same in Table 3. The figures in square brackets are the results of paired t-tests between our DEPSO and its competitors. The t-test result in each case will be one if our DEPSO outperforms its competitor with 95% certainty and zero otherwise. The final result shown in square brackets is the sum of all paired comparison results w.r.t. each test function. Obviously, larger values of the t-test result, with the largest value 4, mean better performance of our DEPSO.

Table 3 Statistical results of employing eight optimizers to minimize 30D benchmark functions

Optimizers	F_1	F_2	F_3	F_4	F_5
DE/*rand*/1/*bin*	(1.1e−05)± (2.2e−06)	(7.6e−06)± (5.1e−07)	(4.0e−07)± (4.0e−07)	(2.9e−02)± (6.0e−04)	(2.2e+01)± (1.8e+00)
Runtime (s)	10.7±0.1	16.1±0.0	7.8±0.1	20.1±1.1	7.0±0.0
SPSO2007	(2.1e−08)± (2.3e−09)	(4.8e−08)± (5.5e−09)	(2.3e−14)± (1.6e−14)	(1.1e−01)± (5.1e−02)	(4.9e+01)± (8.4e+00)
Runtime (s)	11.6±0.1	16.0±0.0	11.4±0.8	17.4±0.6	8.5±0.0
DEPSO-ZX	**(1.1e−13)**± **(8.2e−15)**	(1.1e+00)± (2.7e−01)	(5.4e−12)± (3.7e−12)	(7.9e+00)± (1.2e+00)	(4.5e−01)± (2.1e−01)
Runtime (s)	10.2±0.4	12.9±0.1	8.3±0.4	13.7±0.1	6.2±0.1
Our DEPSO	**(7.7e−15)**± **(5.8e−17)**[4]*	**(7.5e−15)**± **(3.7e−16)**[4]*	**(1.1e−16)**± **(4.9e−17)**[4]*	**(1.6e−03)**± **(2.1e−04)**[4]*	**0**±**0**[4]*
Runtime (s)	10.8±0.2	15.1±0.3	9.1±0.6	16.6±0.4	7.7±0.0
	F_6	F_7	F_8	F_9	F_{10}
DE/*rand*/1/*bin*	**(4.7e+01)**± **(7.2e+00)***	(2.9e+03)± (1.9e+02)	(1.5e+02)± (1.3e+02)	(1.2e+09)± (5.1e+07)	(4.1e−11)± (7.5e−12)
Runtime (s)	16.4±0.0	17.6±0.1	84.3±5.7	15.0±0.9	10.7±0.0
SPSO2007	(1.2e+02)± (6.7e+00)	(2.8e+03)± (1.4e+02)	(7.2e+02)± (8.4e+01)	**(6.1e+01)**± **(1.0e+01)***	(7.8e−16)± (1.5e−16)
Runtime (s)	17.3±0.0	22.2±0.0	94.0±7.6	15.8±0.7	13.1±0.0
DEPSO-ZX	(9.4e+01)± (4.1e+00)	(7.4e+02)± (4.3e+01)	(9.1e+02)± (1.3e+02)	(4.6e+03)± (3.3e+03)	(1.2e+00)± (8.2e−01)
Runtime (s)	15.3±0.1	16.6±0.1	78.5±4.9	13.6±0.7	10.4±0.1
Our DEPSO	**(5.5e+01)**± **(4.4e+00)**[2]	**(8.2e−05)**± **(1.6e−14)**[4]*	**(8.2e−05)**± **(2.0e−14)**[4]*	**(6.3e+01)**± **(5.1e+00)**[2]*	**(1.3e−31)**± **(0)**[4]*
Runtime (s)	16.4±0.1	19.8±0.0	85.3±5.2	15.1±0.8	11.5±0.0

In the optimization of 10D Rotated & Shifted Rastrigin function (F_6), the four optimizers perform comparatively. In the remaining cases, our DEPSO is the best optimizer, and its advantage over other optimizers is prominent especially in the optimization of 30D problems. The experimental results also validate the scalability of our DEPSO since the dimensional increase from $D = 10$ to $D = 30$ almost has no effect on the advantages of this optimizer over its competitors. In addition, the rotation of function landscapes indeed increases the optimization hardness of Alpine function (F_4 versus F_3) and Rastrigin function (F_6 versus F_5) for most optimizers. Nevertheless, our DEPSO remains to be the best optimizer or performs comparatively against the best one in the optimization of rotated functions.

Regarding the time cost (i.e., running time), it can be observed from Tables 2 and 3 that the proposed DEPSO does not cause too much extra computation cost in contrast to other optimizers, which coincides with the observation in our previous research [1].

5 Conclusions

The novel DEPSO optimizer is further confirmed to have excellent performance in the global optimization of multimodal functions including those rotated benchmark functions. The proposed DEPSO has few parameters to be tuned except for its population size and length of learning period. It was observed that the DEPSO is insensitive to the length of its learning period. The statistical learning adopted in the optimizer leads to the adaptation of evolution methods which is very efficient for the hybridization of different optimizers. Fitter evolution methods at different stages can be selected online automatically.

Acknowledgements

This work was supported by the National Science Fund for Distinguished Young Scholars (Grant No. 60925011), and the Open Issues Foundation of the Key Laboratory of Complex System and Intelligence Science, Institute of Automation, Chinese Academy of Sciences (Grant No. 20060104). The authors thank anonymous reviewers and editors for their very useful comments to improve this paper.

References

1 Chen J, Xin B, Peng Z H, et al. Statistical learning makes the hybridization of particle swarm and differential evolution more efficient—a novel hybrid optimizer. Sci China Ser F-Inf Sci, 2009, 52: 1278–1282
2 Clerc M, Kennedy J. The particle swarm: explosion, stability and convergence in a multi-dimensional complex space. IEEE Trans Evol Comput, 2002, 6: 58–73
3 Price K, Storn R M, Lampinen J A. Differential Evolution: a Practical Approach to Global Optimization (Natural Computing Series). New York: Springer, 2005
4 Kennedy J, Mendes R. Population structure and particle swarm performance. In: Proceedings of the World Congress on Computational Intelligence, Honolulu, HI, USA, 2002. 1671–1676
5 Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput, 2004, 8: 240–255
6 Hsieh S T, Sun T Y, Liu C C, et al. Efficient population utilization strategy for particle swarm optimizer. IEEE Trans Syst Man Cybern - Part B: Cybern, 2009, 39: 444–456
7 Janson S, Middendorf M. A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans Syst Man Cybern–Part B: Cybern, 2005, 35: 1272–1282
8 Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal funcions. IEEE Trans Evol Comput, 2006, 10: 281–295
9 Mendes R, Kennedy J. The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput, 2004, 8: 204–210
10 Bergh F van den, Engelbrecht A P. A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput, 2004, 8: 225–239
11 Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer. In: Proceedings of the IEEE Swarm Intelligence Symposium, Pasadena, California, USA, 2005. 124–129
12 Zhang W J, Xie X F. DEPSO: hybrid particle swarm with differential evolution operator. In: Proceedings of the IEEE International Conference on System, Man, Cybernetics, Washington, DC, USA, 2003. 3816–3821
13 Juang C F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern - Part B: Cybern, 2004. 34: 997–1006
14 Chakraborty U K. Advances in Differential Evolution. Berlin: Springer-Verlag, 2008
15 Chen J, Xin B, Peng Z H, et al. Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization. IEEE Trans Syst Man Cybern–Part A: Syst & Human, 2009, 39: 680–691
16 Eiben A E, Hinterding R, Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Trans Evol Comput, 1999, 3: 124–141
17 Sutton A M, Whitley D, Lunacek M, Howe A. PSO and multi-funnel landscapes: how cooperation might limit exploration. In: Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Seattle, Washington, USA, 2006. 75–82
18 Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: a survey of some theoretical and practical aspects of genetic algorithms. BioSystems, 1996, 39: 263–278
19 Liang J J, Suganthan P N, Deb K. Novel composition functions for numerical global optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium, Pasadena, California, USA, 2005. 68–75
20 Brest J, Greiner S, Bošković B, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput, 2006, 10: 646–657