Information Sciences 223 (2013) 119–135

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Diversity enhanced particle swarm optimization with neighborhood search

Hui Wang [a,*], Hui Sun [a], Changhe Li [b], Shahryar Rahnamayan [c], Jeng-shyang Pan [a,d,e]

[a] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, PR China
[b] School of Computer, China University of Geosciences, Wuhan 430072, PR China
[c] Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON, Canada L1H 7K4
[d] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, PR China
[e] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan

ARTICLE INFO

Article history:
Received 27 December 2011
Received in revised form 23 June 2012
Accepted 10 October 2012
Available online 25 October 2012

Keywords:
Particle Swarm Optimization (PSO)
Diversity
Neighborhood search
Global optimization

ABSTRACT

Particle swarm optimization (PSO) has shown an effective performance for solving variant benchmark and real-world optimization problems. However, it suffers from premature convergence because of quick losing of diversity. In order to enhance its performance, this paper proposes a hybrid PSO algorithm, called DNSPSO, which employs a diversity enhancing mechanism and neighborhood search strategies to achieve a trade-off between exploration and exploitation abilities. A comprehensive experimental study is conducted on a set of benchmark functions, including rotated multimodal and shifted high-dimensional problems. Comparison results show that DNSPSO obtains a promising performance on the majority of the test problems.

1. Introduction

In the past decades, several variant swarm intelligence-based algorithms have been proposed to solve complex benchmark and real-world optimization problems, e.g., Particle Swarm Optimization (PSO) [29], Ant Colony Optimization (ACO) [14], Artificial Bee Colony (ABC) [27], Cat Swarm Optimization [7], etc. Due to PSO's simple concept, easy implementation yet effectiveness, it has become popular in evolutionary optimization community.

In PSO, each particle in the population (swarm) flies to its previous best position (*pbest*) and the global best position (*gbest*). Although, the mechanism of this motion can result a fast convergence rate, it suffers from the premature convergence problem, which means it can be easily trapped into local minima when solving multimodal problems. When particles move toward the direction of *pbest* and *gbest*, the dissimilarities (diversity) among particles gradually decreases. When this happens, the search is restricted to a small search space containing these similar particles. Consequently, finding new candidate solutions is very difficult. If reducing diversity takes place too early, a premature stagnation will be caused.

It is known that the performance of PSO is highly related to diversity of particles, specially when attempts are made to avoid premature convergence and to escape from local optima. So, maintaining a higher diversity in PSO is a crucial task. Diversity measures are traditionally used to analyze evolutionary algorithms (EAs) rather than guiding them. Recently, Ursem [51] proposed a diversity-guided evolutionary algorithm, which uses a diversity measure to alternate between exploring and exploiting behaviors. After that some similar diversity-guided strategies are utilized in PSO to enhance its performance [24,40,42,47,54].

* Corresponding author. Tel.: +86 0791 88126661; fax: +86 0791 88126660.
E-mail addresses: wanghui_cug@yahoo.com.cn (H. Wang), sunhui2006@yahoo.com.cn (H. Sun), cl160@mcs.le.ac.uk (C. Li), shahryar.rahnamayan@uoit.ca (S. Rahnamayan), jspan@cc.kuas.edu.tw (J.-s. Pan).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.10.012

Majority of the diversity guided PSO algorithms are usually based on monitoring the diversity of swarm. When the diversity drops below a predefined constant value, a diversity enhancing operator (such as repulsion mechanism) is applied to avoid premature convergence. When the diversity increases to another predefined constant value, a diversity decreasing operator (such as greedy selection) is used to support a fast convergence rate. However, the calculation of diversity is a time-consuming task. In order to maintain diversity and avoid calculating the diversity, this paper proposes a novel diversity enhancing mechanism for PSO. Furthermore, a neighborhood search strategy is employed to improve the local and global search abilities. Experimental studies on 45 well-known benchmark functions, including rotated multimodal and shifted high-dimensional problems, shows that our approach obtains a promising performance.

The rest of the paper is organized as follows. Section 2 presents some PSO related works. Section 3 describes our proposed approach. Section 4 presents experimental simulations, results, and discussions. Finally, the work is concluded in Section 5.

2. Related works

PSO is a population-based stochastic algorithm that starts with an initial population of randomly generated particles. For a search problem in a D-dimensional space, a particle represents a potential solution presented by their velocity and position. During a search process, each particle is attracted by its previous best particle (*pbest*) and the global best particle (*gbest*) as follows [43].

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot rand1_{ij} \cdot (pbest_{ij}(t) - x_{ij}(t)) + c_2 \cdot rand2_{ij} \cdot (gbest_j(t) - x_{ij}(t)), \tag{1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{2}$$

where $i = 1, 2, \ldots, N$ is the particle's index, N is the population size, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is the position of the ith particle; $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ represents the velocity of the ith particle; the $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$ is the best previous position yielding the best fitness value for the ith particle; and $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$ is the global best particle found by all particles so far. The parameter w, called inertia factor, which is used to balance the global and local search abilities of particles [43], $rand1_{ij}$ and $rand2_{ij}$ are two uniform random numbers generated independently within the range of $[0,1]$, c_1 and c_2 are two learning factors which control the influence of the social and cognitive components, and $t = 1, 2, \ldots$ indicates the iteration number.

Since PSO was introduced, it has become a popular optimizer and has widely been applied in practical problems. In the past decades, many variants of PSO have been proposed. A brief overview of these variants is presented as follows.

Shi and Eberhart [43] introduced a parameter called inertia weight w for the classical PSO. The inertia weight is used to balance the exploration and the exploitation abilities, a linearly decreasing w over the search process was a good choice [43]. Clerc and Kennedy [8] proposed a constriction factor, which can guarantee the convergence and improve the convergence velocity. Bergh and Engelbrecht [2] presented a comprehensive study on the parameters of PSO, and provided a formal proof that each particle converges to a stable point.

In order to improve the performance of PSO, different types of topologies have been proposed. Kennedy [28,30] analyzed the effects of neighborhood topology on PSO, and proposed four different neighborhood topologies. The presented results show that PSO with a small neighborhood may perform better on complex problems, while PSO with a large neighborhood may perform better on simple problems. Suganthan [45] introduced a neighborhood operator which gradually increases the neighborhood size of a particle until it covers all particles in a swarm. Hu and Eberhart [25] updated the neighborhood of each particle by dynamically selecting m particles that are the nearest to the current particle. Mendes and Kennedy [37] proposed a fully informed PSO algorithm (FIPS), in which the neighbors of each particle, instead of *pbest* and *gbest*, are used to update the velocity. Based on the FIPS, Mohais et al. [38] presented random and dynamic neighborhoods by re-structuring neighborhoods in terms of a diversity-preserving measure. Peram et al. [41] presented a modified PSO called fitness-distance-ration-based PSO (FDR-PSO), which employs a new velocity updating method. Yano et al. [58] proposed a hybrid PSO algorithm with a neighborhood search. When the current position of a particle is better than its previous best position, the algorithm will search $2^D - 1$ points in the neighborhood of the current point (D is the dimension size). Nevertheless, the number of search points exponentially grow with the increase of D. To tackle this problem, they defined a maximum number of search points for different kinds of problems. However, the algorithm converges very slowly because the neighborhood search is time consuming.

Bergh and Engelbrecht [1] proposed a cooperative approach for PSO (CPSO-H) for solving multimodal problems. Liang et al. [34] introduced a comprehensive learning PSO (CPSO) to learn other particles' experiences in different dimensions. Chen et al. [6] proposed a novel PSO algorithm by employ dynamic linkage discovery and recombination to improve the performance of PSO. The former is a costless and effective linkage recognition technique, and the latter utilizes the discovered linkage configuration to promote the cooperation of PSO. Ho et al. [22] introduced an orthogonal method for PSO to solve task assignment problems. Li and Yang [31,32] presented an adaptive learning PSO for function optimization, in which the learning mechanism of each particle is separated into three components: its own historical best position, the closest neighbor and the global best one. By using this individual level adaptive technique, a particle can well control its well-balanced behavior of exploration and exploitation. Hsieh et al. [23] presented an efficient population utilization strategy for PSO (EPUS-PSO), which introduced a population manager and a solution sharing strategies. In EPUS-PSO, the population size is

variable. The population manager can increase or decrease particle numbers according to the searching status. Cervantes et al. [4] proposed an adaptive Michigan PSO (AMPSO) to reduce the dimension of the search space. The reported results confirm that AMPSO is able to improve the performance of the nearest neighbor classifiers. Zhan et al. [59] presented an adaptive PSO (APSO) by employing two following strategies. The first one evaluates the population distribution and particle fitness and identifies the current search status. The second one utilizes an elitist learning strategy to help the global best particle jump out of the likely local optima.

Sun et al. [48] proposed an improved vector PSO (IVPSO) to solve constrained optimization problems. Wang et al. [56] introduced a self-adaptive learning strategy to improve the performance of CLPSO. Wang et al. [55] proposed another improved CLPSO by employing a generalized opposition-based learning (GOBL). Hsul et al. [26] presented a novel PSO algorithm for the multiple interference cancellation design of a linear phase array. To overcome the drawbacks of Performance-dependent PSO [3], four different chaotic sequence are incorporated to enhance the exploration ability [9]. Chakraborty et al. [5] presented a simple analysis of general multi-objective PSO. The results demonstrate that the inertial factor and acceleration coefficients control the convergence behavior of the algorithm to the Pareto front in the objective function space. In [10], PSO is regarded as a two-inputs and one-output feedback system. Two PID controllers are incorporated into the methodology of PSO to improve the diversity of swarm. Wang et al. [53] introduced generalized opposition-based learning and Cauchy mutation to improve the performance of PSO. Simulation results demonstrate that the proposed approach (GOPSO) obtains a promising performance for low-dimensional problems ($D \leqslant 30$). However, GOPSO is not a good choice to solve high-dimensional problems ($D = 100$). Lu et al. [36] proposed a new hybrid PSO algorithm, in which a real-valued mutation (RVM) operator was embedded into three variants of PSO algorithms. In [33], Li et al. proposed a modified Broyden–Fletcher–Goldfarb–Shanno (BFGS) method which was integrated into the context of PSOs to enhance local search ability. Simulation results demonstrated that the BFGS could effectively improve the performance of many PSO variants.

3. Diversity enhanced PSO with neighborhood search (DNSPSO)

In this section, a new PSO variant called diversity enhanced PSO with neighborhood search (DNSPSO) is proposed. The DNSPSO employs two strategies including diversity enhanced mechanism and neighborhood search.

3.1. Diversity enhancing mechanism

Diversity loss is a serious problem for PSO algorithms during the search process. To maintain diversity of a swarm, some excellent diversity enhancing strategies have been proposed. Riget [42] proposed a diversity-guided PSO, called ARPSO, which defines a repulsion phase based on a modified velocity updating model. The basic PSO algorithm only employs an attraction phase, in which particles are attracted by their *pbests* and the *gbest*. All particles in the swarm move quickly to the same direction and the similarities among particles increase very fast. As a result, the diversity of the swarm decreases. To avoid particles attracting too fast toward the best particles, a repulsion phase is introduced. When the diversity of swarm drops below a predefined constant number d_{low}, the ARPSO switches to the repulsion phase, in which particles repel each other, and then the diversity increases. When the diversity reaches to a high level (d_{high}), the ARPSO switches back to the attraction phase. The whole search process of ARPSO alternates between phase of exploiting and exploring - attraction and repulsion - lower and higher diversities. However, ARPSO does not change the search behavior when the diversity remains between d_{low} and d_{high}. Based on the ARPSO, Pant et al. [40] proposed a middle phase (called positive conflict phase) between attraction and repulsion. In the middle phase, there is neither a complete attraction nor a complete repulsion. Each particle is attracted by its previous best particle and is repelled by the global best particle. In [47], Sun et al. introduced a mutation operator to enhanced the swarm diversity. When the diversity is below d_{low}, a mutation is conducted on *gbest* to increase the dissimilarities among particles, *pbest* and *gbest*.

Although the abovementioned diversity enhancing strategies have a good performance, they cost much computational time to monitor and calculate the swarm diversity.

$$Diversity(t) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{D}\left(x_{ij}(t) - \overline{x_j(t)}\right)^2}, \tag{3}$$

$$\overline{x_j(t)} = \frac{\sum_{i=1}^{N} x_{ij}(t)}{N}. \tag{4}$$

To avoid the diversity calculation and maintain the swarm diversity, this paper presents a novel diversity enhanced mechanism. For each particle $P_i(t)$, a new particle $P_i(t+1)$ is generated by the PSO's velocity and position updating equations. By recombining $P_i(t)$ and $P_i(t+1)$, a trial particle $TP_i(t+1) = (TX_i(t+1), TV_i(t+1))$ is generated as follows:

$$TX_{ij}(t+1) = \begin{cases} X_{ij}(t+1), & \text{if } rand_j(0,1) < p_r \\ X_{ij}(t), & \text{otherwise} \end{cases}, \tag{5}$$

$$TV_{ij}(t+1) = V_{ij}(t+1), \tag{6}$$

where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, $rand_j(0,1)$ is a uniform random number within $[0,1]$, and p_r is a predefined probability.

Each element of the position vector in $TP_i(t+1)$ inherits the elements from $X_i(t+1)$ and $X_i(t)$ with probabilities of p_r and $1 - p_r$, respectively. This is similar to the crossover operation of Differential Evolution (DE) [44]. A clear visualization of how to generate $TP_i(t+1)$ is presented in Fig. 1. It can be seen from Eq. (5) that a smaller p_r will increase the dissimilarities between $TP_i(t+1)$ and $P_i(t+1)$. For an extreme case with $p_r = 0$, $TP_i(t+1)$ is equivalent to $P_i(t)$. A larger p_r will increase the similarities between $TP_i(t+1)$ and $P_i(t+1)$. For $p_r = 1$, $TP_i(t+1)$ is equal to $P_i(t+1)$.

After the recombination, a greedy selection mechanism is used as follows:

$$P_i(t+1) = \begin{cases} TP_i(t+1), & \text{if } f(TP_i(t+1)) \leqslant f(P_i(t+1)) \\ P_i(t+1), & \text{otherwise} \end{cases}, \tag{7}$$

where $f(\cdot)$ is the fitness evaluation function. Without loss of generality, this paper only considers minimization problems. If, and only if, the trial particle $TP_i(t+1)$ is better than $P_i(t+1)$, then replace $P_i(t+1)$ with $TP_i(t+1)$; otherwise, the $P_i(t+1)$ remains unchanged.

In PSO, particles tend to move the same position during the search process. It means that particles become similar with increasing of iterations. When the trial particle $TP_i(t+1)$ is selected into the next generation, the dissimilarities between $TP_i(t+1)$ and $P_i(t+1)$ will determine the dissimilarities among $P_i(t+1)$ (replaced by $TP_i(t+1)$) and the rest particles of swarm. Larger dissimilarities in the swarm means a higher diversity. Therefore, the value of p_r controls the swarm diversity. A smaller p_r makes larger dissimilarities between $TP_i(t+1)$ and $P_i(t+1)$ resulting a higher diversity, while a larger p_r will decrease the diversity.

Some similar algorithms have been proposed by hybridization of PSO and the DE schemes, such as [15,16,60]. However, our approach differs from them. In [15,16], DE algorithm was used to evolve the pervious best particles (*pbests*) to enhance the convergence. In [60], mutations provided by DE algorithm were conducted on the *pbest*.

3.2. Neighborhood search strategy

Like other stochastic search algorithms, PSO also suffers from the problem of premature convergence when solving highly multimodal problems. Sometimes, the suboptima are near to the global optimum and the neighborhoods of trapped individuals may contain the global optimum. At such situation, searching the neighborhoods of individuals is helpful to find better solutions. Based on this idea, some excellent neighborhood search strategies have been applied to some nature-inspired algorithms [11,35,37,57].

Kennedy designed four different population topologies, including circle, wheel, star, and random [28]. The reported results shows that population topologies with fewer connections might perform better on highly multimodal problems, while highly interconnected populations would be better for unimodal problems. In [30], Kennedy and Mendes investigated the effects of various population topologies on PSO to seek a better structure that performs well on a variety of test problems. Suganthan [45] proposed a variable neighborhood operator. During the initial states of the optimization, the neighborhood is an individual particle itself. As the number of generation increases, the neighborhood is gradually extended to include all particles. Mendes et al. [37] proposed a fully informed PSO algorithm (FIPS), in which the neighbors of each particle, instead of *pbest* and *gbest*, are used to update the velocity. Peram et al. [41] developed the fitness-distance-ratio based PSO (called FDR-PSO), in which each particle is attracted towards the best previous positions visited by its neighbors. Hu and Eberhart [25] used dynamic neighborhood PSO to solve multi-objective optimization problems. In each generation, after calculating distance to every other particle, each particle finds its new neighbors. Among the new neighbors, each particle finds the local best particle as the *lbest*. Ghosh et al. [20] presented a probabilistic analysis of the particle interaction and information exchange in an *lbest* PSO with variable random neighborhood topology.

Always, we are looking for a tradeoff between exploration and exploitation for EAs. The former indicates the global search ability and makes the algorithm explore every region of the feasible search space, while the latter means the local search ability, and accelerates the algorithm converging to the near-optimal solutions. Most improvements on EAs try to seek a

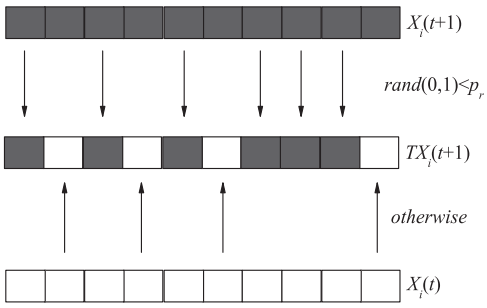


Fig. 1. The method of generating a trial particle $TP_i(t+1)$.

desirable balance between these two phases to suit different kinds of problems. To tackle this problem, this paper presents one local and one global neighborhood search strategies, which are inspired by the neighborhood mutation operator in Differential Evolution (DE) [11].

Assume that P_i, $i = 1, 2, \ldots, N$ is the ith particle in the swarm, where N is the population size. The N particles are organized on a circle topology according to their indices, such that P_N and P_2 are the two immediate neighbors of P_1 [11]. Fig. 2 presents a circle topology, where there are 16 particles in the swarm. Although there are various neighborhood topologies (like star, wheel, pyramid, 4-clusters, and circular) for PSO [28], the ring topology is simple and easy to implement. For each particle P_i, its k-neighborhood radius consisting of particles $P_{i-k}, \ldots, P_i, \ldots, P_{i+k}$, where k is a integer $0 \leqslant k \leqslant \frac{N-1}{2}$. Fig. 2 shows the k-neighborhood radius, where $k = 2$. According to our empirical studies, the parameter k does not affect the performance of DNSPSO. Different values of k obtain similar performance. In this paper, k is set to 2.

3.2.1. Local neighborhood search (LNS) strategy

For each particle, its neighborhood may cover better solutions. To improve the ability of exploitation, a local neighborhood search (LNS) strategy is proposed. During searching the neighborhood of a particle P_i, a trial particle $L_i = (LX_i, LV_i)$ is generated as follows [52]:

$$LX_i = r_1 \cdot X_i + r_2 \cdot pbest_i + r_3 \cdot (X_c - X_d), \tag{8}$$

$$LV_i = V_i, \tag{9}$$

where X_i is the position vector of the ith particle, $pbest_i$ is the previous best particle of P_i, X_c and X_d are the position vectors of two random particles in the k-neighborhood radius of P_i, $c, d \in [i-k, i+k] \wedge c \neq d \neq i$, r_1, r_2 and r_3 are three uniform random numbers within (0,1), and $r_1 + r_2 + r_3 = 1$. The random numbers r_1, r_2 and r_3 are the same for all $j = 1, 2, \ldots, D$, and they are generated anew in each generation. A clear explanation of the mechanism of LNS is presented in Fig. 3a. The $pbest_i$ is the previous best particle of X_i, so it is not on the circle topology. To keep the flying direction of P_i, the trial particle L_i keeps the same velocity of P_i.

The local neighborhood search strategy is effective when the local minima is near to the global optimum. Particles with large jumps are easily trapped in local optima. The LNS could generate a trial particle near to the current search point. This is helpful to find the global optimum step-by-step.

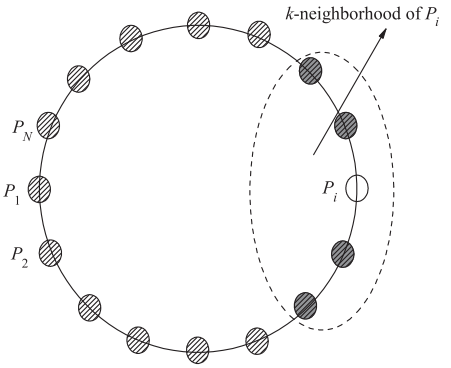
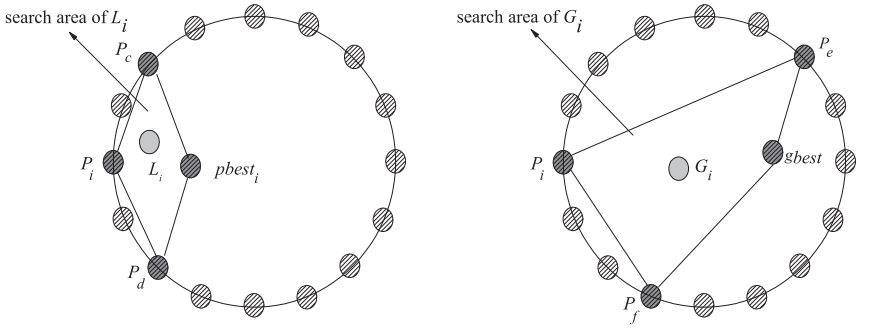


Fig. 2. The circle topology and k-neighborhood.



(a) The local neighborhood search (LNS) (b) The global neighborhood search (GNS)

Fig. 3. The local and global neighborhood search strategies.

3.2.2. Global neighborhood search (GNS) strategy

Besides the LNS, a global neighborhood search (GNS) strategy is proposed to enhance the ability of exploration. When searching the neighborhood of a particle P_i, another trial particle $G_i = (GX_i, GV_i)$ is generated as follows [52]

$$GX_i = r_4 \cdot X_i + r_5 \cdot gbest + r_6 \cdot (X_e - X_f), \tag{10}$$

$$GV_i = V_i, \tag{11}$$

where *gbest* is the global best particle, X_e and X_f are the position vectors of two random particles chosen for the entire swarm, $e, f \in [1, N] \wedge e \neq f \neq i$, r_4, r_5 and r_6 are three uniform random numbers within (0,1), and $r_4 + r_5 + r_6 = 1$. The random numbers r_4, r_5 and r_6 are the same for all $j = 1, 2, \ldots, D$, and they are generated anew in each generation. The GNS strategy is helpful to solve multimodal problems. Particles are located at different regions. Therefore, if the current particle falls into local minima, particles in other regions may pull the trapped particle forward.

For both LNS and GNS strategies, the newly generated trial particles, L_i and G_i, keep the same velocity of their parent P_i. This aims to follow the same flying direction and jumping size of their parent.

3.3. The proposed approach

The DNSPSO algorithm employs two strategies including diversity enhancing mechanism and neighborhood search operators. The former aims to indirectly enhance the swarm diversity, and the latter focuses on searching the neighbors of particles.

In every generation, a trial particle TP_i is generated by the recombination operator, Eqs. (eqdiv1) and (6). If TP_i is better than its parent P_i, then replace P_i with TP_i; otherwise, we keep P_i unchanged. After this operation, the neighborhood search strategy is conducted with p_{ns} probability. If the probability p_{ns} is satisfied, two trial particles, L_i and G_i, are generated. Then, the fittest particle among P_i, L_i and G_i is selected as the new P_i.

Algorithm 1. The Proposed DNSPSO Algorithm

```
1  Uniformly randomly initialize each particle in the swarm;
2  Initialize pbest and gbest;
3  while FEs ⩽ MAX_FEs do
4    for i = 1 to N do
5      Calculate the velocity of particle P_i according to Eq. (1);
6      Update the position of particle P_i according to Eq. (2);
7      Calculate the fitness value of P_i;
8      FEs++;
       /* Diversity enhanced mechanism */
9      Generate a new trial particle TP_i according to Eqs. (5) and (6);
10     Calculate the fitness value of TP_i;
11     FEs++;
12     Select a fitter one between P_i and TP_i as the new P_i (see Eq. (7));
13     Update pbest_i and gbest;
14   end
15   for i = 1 to N do
       /* Neighborhood search strategy */
16     if rand(0,1) ⩽ p_ns then
17       Generate a trial particle L_i according to Eqs. (8) and (9);
18       Generate a trial particle G_i according to Eqs. (10) and (11);
19       Calculate the fitness values of L_i and G_i;
20       FEs = FEs + 2;
21       Select the fittest one among P_i, L_i and G_i as the new P_i;
22     end
23     Update pbest_i and gbest;
24   end
25 end
```

The main steps of DNSPSO are described in Algorithm 1, where P_i is ith particle in the swarm, N is the population size, p_{ns} is the probability of conducting neighborhood search, FEs is the number of fitness evaluations, and MAX_FEs is the maximum number of function evaluations.

4. Experimental verifications

4.1. Test problems

There are 15 benchmark functions used in the following experiments. These problems were utilized in previous studies [34,53]. According to their properties, they are divided into four classes: unimodal and simple multimodal problems ($f_1 - f_2$), unrotated multimodal problems ($f_3 - f_7$), rotated multimodal problems ($f_8 - f_{11}$), and high-dimensional shifted problems ($f_{12} - f_{15}$). All the problems used in this paper are minimization problems. The brief descriptions of these benchmark problems are listed in Table 1. More details about the definition of benchmark problems can be found in [34,50].

4.2. Parameter sensitivity analysis

The values of p_r and p_{ns} may affect the performance of DNSPSO. To select better values of these two control parameters, we investigated the performance of DNSPSO under variant p_r and p_{ns} values. In this section, p_r and p_{ns} are selected from the set $\{0.0, 0.3, 0.6, 0.9\}$ and $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, respectively. Therefore, there are $4 \times 6 = 24$ possible choices for the combination of p_r and p_{ns}.

For other parameters of DNSPSO, we used the following settings. The population size N was set to 40. $w = 0.7298$, $c_1 = c_2 = 1.49618$, $k = 2$. For $D = 30$, the maximum number of FEs (MAX_FEs) was set to 200,000 [34]. For $D = 100$, MAX_FEs $= 5000 \times D$ [50]. All the experiments were conducted 30 times, and the mean error values $f(x) - f(x^o)$ ($f(x^o)$ is the global optimum of $f(x)$) were recorded.

For different choices of the mentioned parameters, there are 24 groups of results in total. To simplify these results, the relatively good parameter settings are presented in Table 2. From the results, it can be seen that a smaller p_r is suitable for low-dimensional problems ($f_1 - f_{11}$), while a larger p_r is helpful to solve high-dimensional problems ($f_{12} - f_{15}$). As mentioned before, a smaller p_r increases the diversity. This slows down the convergence rate of DNSPSO. That is one of the possible reasons that a smaller p_r performs bad on high-dimensional problems. Though the performance of DNSPSO is not sensitive to the value of p_{ns}, it does not mean that the neighborhood search does not work properly during the searching process. According to our experiments, the performance of DNSPSO is very poor when $p_{ns} = 0.0$. Even if p_{ns} is set to 0.05, DNSPSO also works very well.

To select the best group of parameter settings at a statistical level, average ranking of Friedman test is conducted according to the suggestions of [13,17]. Table 3 shows the average rankings of DNSPSO with different parameter settings. As shown, $p_r = 0.9$, $p_{ns} = 0.6$ achieves the highest ranking. It demonstrates that $p_r = 0.9$ and $p_{ns} = 0.6$ are the relatively best choices for the test suite.

4.3. Effects of different strategies

The proposed approach employs two strategies: diversity enhancing mechanism and neighborhood searching. To investigate the effects of these two strategies, we studied the performance of standard PSO, PSO with diversity enhanced mechanism (DPSO), PSO with neighborhood search (NSPSO), and DNSPSO which includes both mechanisms. This will help to verify the effectiveness of these strategies separately.

For the parameters p_r and p_{ns} used in DPSO, NSPSO and DNSPSO, they are set to 0.9 and 0.6, respectively. For other parameters, we have used the same settings as descried in the Section 4.2.

Table 1
Benchmark problems used in the experiments, where D is the dimension of the functions, $X \in R^D$ is the definition domain, and $f(x^o)$ is the global optimum of the function.

Problems	Name	D	X	$f(x^o)$
f_1	Sphere function	30	$[-100, 100]$	0
f_2	Rosenbrock's function	30	$[-2.048, 2.048]$	0
f_3	Ackley's function	30	$[-32.768, 32.768]$	0
f_4	Griewanks's function	30	$[-600, 600]$	0
f_5	Weierstrass function	30	$[-0.5, 0.5]$	0
f_6	Rastrigin's function	30	$[-5.12, 5.12]$	0
f_7	Nocontinuous Rastrigin's function	30	$[-5.12, 5.12]$	0
f_8	Rotated Ackley's function	30	$[-32.768, 32.768]$	0
f_9	Rotated Griewanks's function	30	$[-600, 600]$	0
f_{10}	Rotated Weierstrass function	30	$[-0.5, 0.5]$	0
f_{11}	Rotated Rastrigin's function	30	$[-5.12, 5.12]$	0
f_{12}	Shifted Sphere function	100	$[-100, 100]$	−450
f_{13}	Shifted Schwefel's function 2.21	100	$[-100, 100]$	−450
f_{14}	Shifted Rosenbrock's function	100	$[-100, 100]$	390
f_{15}	Shifted Griewanks's function	100	$[-600, 600]$	−180

Table 2
Mean error values achieved by DNSPSO with different p_r and p_{ns}. The best results among the nine groups of parameter settings are shown in bold.

Functions	D	$p_r = 0.0$, $p_{ns} = 0.2$ Mean error	$p_r = 0.0$, $p_{ns} = 0.4$ Mean error	$p_r = 0.0$, $p_{ns} = 0.6$ Mean error	$p_r = 0.3$, $p_{ns} = 0.8$ Mean error	$p_r = 0.6$, $p_{ns} = 0.8$ Mean error	$p_r = 0.9$, $p_{ns} = 0.2$ Mean error	$p_r = 0.9$, $p_{ns} = 0.4$ Mean error	$p_r = 0.9$, $p_{ns} = 0.6$ Mean error	$p_r = 0.9$, $p_{ns} = 0.8$ Mean error
f_1	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	30	**1.31E+01**	1.53E+01	1.83E+01	2.49E+01	2.24E+01	1.93E+01	2.05E+01	2.01E+01	2.06E+01
f_3	30	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**
f_4	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_5	30	6.48E−13	3.27E−13	5.34E−13	1.61E−13	1.59E−13	2.04E−13	1.68E−13	**1.40 E−13**	2.63E−13
f_6	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.40E+01	3.84E−01	5.98E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_8	30	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**	**5.89E−16**
f_9	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{10}	30	2.35E−13	**2.14 E−13**	4.38E−13	3.98E+01	2.65E+01	3.71E+01	3.76E+01	2.69E+01	3.86E+01
f_{11}	30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	100	6.95E+04	8.71E+04	1.04E+05	4.08E+03	3.79E−01	**3.57 E−13**	1.95E−11	4.24E−11	7.39E−09
f_{13}	100	5.23E+01	5.49E+01	5.39E+01	3.44E+01	3.76E+01	2.43E+01	3.34E+01	**2.38E+01**	2.70E+01
f_{14}	100	3.94E+08	1.38E+09	3.12E+09	1.14E+08	1.00E+06	3.60E+02	2.36E+02	**2.19E+02**	2.78E+02
f_{15}	100	1.08E+02	8.35E+01	4.99E+01	8.29E−02	2.90E−01	1.65E−01	5.46E−04	**2.93 E−06**	1.88E−02

Table 3
Average rankings achieved by Friedman test for DNSPSO with different parameter settings. The highest ranking is shown in bold.

DNSPSO	Rankings
$p_r = 0.0$, $p_{ns} = 0.2$	4.63
$p_r = 0.0$, $p_{ns} = 0.4$	4.57
$p_r = 0.0$, $p_{ns} = 0.6$	4.30
$p_r = 0.3$, $p_{ns} = 0.8$	4.27
$p_r = 0.6$, $p_{ns} = 0.8$	4.73
$p_r = 0.9$, $p_{ns} = 0.2$	5.40
$p_r = 0.9$, $p_{ns} = 0.4$	5.64
$p_r = 0.9$, $p_{ns} = 0.6$	**6.30**
$p_r = 0.9$, $p_{ns} = 0.8$	5.17

Table 4
Mean error values achieved by PSO, NSPSO, DPSO and DNSPSO. The best results among the four algorithms are shown in bold.

Functions	D	PSO mean error	NSPSO mean error	DPSO mean error	DNSPSO mean error
f_1	30	2.95E−92	**0.00E+00**	5.33E−49	**0.00E+00**
f_2	30	2.36E+01	1.84E+01	**1.74E+01**	2.01E+01
f_3	30	1.50E+00	**5.89E−16**	1.34E+00	**5.89E−16**
f_4	30	4.65E−02	**0.00E+00**	1.72E−02	**0.00E+00**
f_5	30	2.98E+00	7.39E−13	1.78E−13	**1.40E−13**
f_6	30	4.28E+01	**0.00E+00**	1.82E+01	**0.00E+00**
f_7	30	1.60E+01	**0.00E+00**	9.27E+00	**0.00E+00**
f_8	30	3.55E+00	9.55E−13	2.89E+00	**5.89E−16**
f_9	30	1.97E−02	**0.00E+00**	1.11E−16	**0.00E+00**
f_{10}	30	6.07E+00	**5.86E−01**	3.22E+01	2.69E+01
f_{11}	30	7.36E+01	**0.00E+00**	6.95E+01	**0.00E+00**
f_{12}	100	3.70E+04	9.17E+04	5.06E+04	**4.24E−11**
f_{13}	100	3.92E+01	3.46E+01	3.81E+01	**2.38E+01**
f_{14}	100	**1.97E+02**	1.22E+07	1.76E+07	2.19E+02
f_{15}	100	7.90E+02	9.04E+01	8.39E+00	**2.93E−06**

Table 4 presents the mean error values of the above four algorithms. From the comparison of NSPSO with PSO, NSPSO achieved better results than PSO in all test cases except for f_{14}. On this function, both NSPSO and DPSO hardly improved the quality of solution. Because f_{14} is a shifted Rosenbrock function and many PSO variants hardly find the global optimum which is inside a long, narrow, parabolic shaped flat valley. It shows the neighborhood search was effective for most test functions. PSO performs better than DPSO on f_1, f_{10}, and f_{14}, while DPSO outperforms PSO for the rest functions. The comparison of DPSO with PSO demonstrates the effectiveness of the diversity enhanced mechanism.

By combining these two strategies, DNSPSO obtains excellent performance. It outperforms other three PSO algorithms on the majority of the test functions. Specially for high-dimensional problems ($f_{12} - f_{15}$), using a single strategy (diversity enhanced mechanism or neighborhood search) can hardly achieve promising solutions. For f_{12} and f_{15}, both NSPSO and DPSO fall into local minima, while DNSPSO achieves satisfactory results. The diversity enhancing mechanism is helpful to increase

the swarm diversity, while it slows down the convergence rate. It can be seen from non-shifted and shifted Sphere functions (f_1 and f_{12}), PSO obtains more accurate solutions compared to DPSO. The neighborhood search is based on the attraction of *pbest* or *gbest*. This helps to accelerate the convergence rate of PSO, but runs the risk of losing swarm diversity. By hybridization of these two strategies, DNSPSO achieves a balance between the exploration and exploitation abilities.

The DNSPSO employs two strategies: diversity enhancing mechanism and neighborhood searching, which may cost additional computational time. To investigate how these strategies affect the computational time, we calculated the mean computational time of PSO, NSPSO, DPSO and DNSPSO on the test suite. The computational configurations are listed as follows.

- System: Windows XP (SP3).
- CPU: Intel Core 2 Duo CPU T6400 (2.00 GHz).
- RAM: 2G.
- Language: C++.
- Compiler: Microsoft Visual C++6.0.
- MAX_FEs: 2.00E + 05 ($D = 30$) and $5000 \times D$($D = 100$).

In the experiments, the same parameter settings are used as described in Section 4.2. The mean computational time of the four algorithms on the test suite are listed in Table 5. As seen, DPSO consumes less time than PSO. The main reason is that DPSO presents less iterations than PSO. In the experiments, all algorithms use the same MAX_FEs. For PSO, the maximum number of generations (MAX_G) is $\frac{\text{MAX\_FEs}}{N}$, while the MAX_G of DPSO is $\frac{\text{MAX\_FEs}}{2*N}$. Because DPSO evaluates twice fitness values every generation. For NSPSO, the MAX_G is $\frac{\text{MAX\_FEs}}{N+N*2*p_{ns}} = \frac{\text{MAX\_FEs}}{2.2*N}$. So, NSPSO consumes less iterations than DPSO and PSO. The results of computational time confirms that NSPSO spends less computational time than PSO and DPSO. For DNSPSO, the MAX_G is $\frac{\text{MAX\_FEs}}{3.2*N}$. Though, DNSPSO spends less iterations than NSPSO, it spends more computational time. The main reason is that the computational complexity of the diversity mechanism is higher than the neighborhood search strategy. The above results demonstrates that the combination of diversity mechanism and neighborhood search do not increase additional computational time for DNSPSO when the same MAX_FEs is used.

4.4. Comparison of DNSPSO with other state-of-the-art PSO variants

4.4.1. Comparison of DNSPSO with CPSO-H, CLPSO and APSO

This section presents a comparative study of DNSPSO with CPSO-H, CLPSO and APSO on the 15 test functions. To verify the generality of our proposed strategies, we integrate the proposed diversity mechanism and neighborhood search strategy into CLPSO. The combination of this new approach was called DNSCLPSO. The involved algorithms are listed as follows:

- Cooperative PSO (CPSO-H) [1].
- Comprehensive learning PSO (CLPSO) [34].
- Adaptive PSO (APSO) [59].
- PSO with generalized opposition-based learning (GOPSO) [53].
- Combining CLPSO with the proposed diversity mechanism and neighborhood search strategy (DNSCLPSO).
- Our approach DNSPSO.

The parameter settings of CLPSO-H and CLPSO are described in [34]. By the suggestions of [59], the same parameter settings of ALPSO are used. For GOPSO and DNSPSO, $w = 0.7298$, $c_1 = c_2 = 1.49618$. The probability of using opposition in GOPSO is set to 0.3. The parameters k, p_r, p_{ns} used in DNSPSO are set to 2, 0.9, and 0.6, respectively. For DNSCLPSO, the same param-

Table 5
Mean computational time (in seconds) achieved by PSO, NSPSO, DPSO and DNSPSO.

Functions	D	PSO time	NSPSO time	DPSO time	DNSPSO time
f_1	30	0.84	0.74	0.83	0.75
f_2	30	0.86	0.70	0.86	0.75
f_3	30	1.29	1.13	1.25	1.14
f_4	30	1.38	1.16	1.28	1.20
f_5	30	43.46	42.17	42.76	43.72
f_6	30	1.26	1.05	1.26	1.14
f_7	30	1.74	1.56	1.73	1.48
f_8	30	2.57	2.32	2.48	2.44
f_9	30	2.60	2.38	2.52	2.74
f_{10}	30	49.60	46.81	48.54	47.06
f_{11}	30	2.45	2.18	2.36	2.35
f_{12}	100	6.88	4.91	6.95	4.95
f_{13}	100	8.06	7.16	8.13	7.80
f_{14}	100	28.92	27.24	31.61	27.72
f_{15}	100	20.70	16.80	19.06	16.89

eter settings with CLPSO and DNSPSO are used. The above six PSO algorithms use the same population size ($N = 40$) and maximum number of fitness evaluations (MAX_FEs). For $D = 30$, MAX_FEs is set to 200,000 [34]. For $D = 100$, MAX_FEs is set to $5000 \times D$ [53]. All the experiments are conducted 30 times, and the mean error values $f(x) - f(x^o)$ ($f(x^o)$ is the global optimum of $f(x)$) and variances are recorded.

Results of mean error values and standard deviations achieved by the six PSO algorithms are listed in Table 6. The comparison results among DNSPSO and other algorithms are summarized as "$w/t/l$" in the last row of the table, which means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors.

From the results of Table 6, DNSPSO outperforms CPSO-H on 12 functions, while CPSO-H only achieves better results on 3 functions. CLPSO only obtains better performance than DNSPSO on f_{15}, while DNSPSO outperforms CLPSO for the rest 14 functions. APSO achieves better results than DNSPSO on two functions, while DNSPSO performs better on the rest 13 functions. Both GOPSO and DNSPSO can find the global optimum on six functions. For the rest 9 functions, DNSPSO wins 7, while GOPSO only wins 2.

All algorithms fall into local minima on shifted and non-shifted Rosenbrock functions (f_2 and f_{14}), whose global optimum is inside a long, parabolic shaped flat valley. Most PSO algorithms could easily find the valley, but hardly converge to the global optimum. For shifted Schwefel function (f_{13}), though DNSPSO achieves better solution than other algorithms, all the five PSO algorithms fall into local minima because of the effects of dimension. For low-dimensional Schwefel function, many PSO algorithms can find promising solutions. As dimension increases, the performance of these algorithms are seriously affected.

By combining CLPSO with our proposed strategies, DNSCLPSO achieves significant improvements on 11 functions. Specially for f_3, f_4, f_9 and f_{10}, DNSCLPSO is able to find the global optimum, while CLPSO falls into local optima. However, our proposed approaches do not always work for all test functions. For f_6 and f_7, CLPSO achieved reasonable solutions, while DNSCLPSO gets stuck. For f_{14}, DNSCLPSO can hardly find good solutions.

Fig. 4 shows the convergence curves of the six PSO algorithms. This paper only presents four representative convergence graphs on high-dimensional functions ($f_{12} - f_{15}$). It can be seen that, for shifted Sphere and shifted Rosenbrock functions, DNSPSO converges faster than other four algorithms during the evolution. For shifted Schwefel function, DNSPSO shows faster convergence speed at the last stage of the evolution. For shifted Griewank function, DNSPSO converges faster at the beginning and the middle stages of the evolution than the other algorithms, while DNSCLPSO achieves a faster convergence than that of the other algorithms at the last stage of the evolution.

In order to compare the performance of multiple algorithms on the test suite, we conduct Friedman test according to the suggestions of [13,18]. Table 7 shows the average ranking of CPSO-H, CLPSO, APSO, GOPSO, DNSCLPSO and DNSPSO. The highest ranking is shown in bold. As seen, the performance of the six algorithms ranks as follows: DNSPSO, DNSCLPSO, GOPSO, CLPSO, CPSO-H, and APSO. The highest average ranking is obtained by the DNSPSO algorithm. It demonstrates that DNSPSO is the best one among the six PSO algorithms.

To compare the performance differences among DNSPSO and the other four PSO algorithms, we conduct a Wilcoxon signed-rank test [12,19]. Table 8 shows the resultant p-values when comparing among DNSPSO and the other five algorithms. The p-values below 0.05 are shown in bold. From the results, it can be seen that DNSPSO is significantly better than all algorithms except for GOPSO and DNSCLPSO. Although DNSPSO is not significantly better than GOPSO and DNSCLPSO, DNSPSO performs better than them according to the average rankings shown in Table 7.

4.4.2. Comparison of DNSPSO with PSO based on real-valued mutation

Recently, Lu et al. [36] proposed a new hybrid PSO algorithm, in which a real-valued mutation (RVM) operator was embedded into three variants of PSO algorithms. Experimental results on six benchmark functions showed that the RVM can efficiently improve the performance of PSO variants. In this section, we present a comparative study of DNSPSO with PSO variants based on RVM. The involved algorithms are listed as follows.

Table 6
Comparison results of mean function error values and standard deviations, where "$w/t/l$" means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	D	CPSO-H [1]	CLPSO [34]	APSO [59]	GOPSO [53]	DNSCLPSO	DNSPSO
f_1	30	1.29E−36 ± 7.61E−36	1.23E−13 ± 3.09E−13	9.60E−66 ± 1.57E−65	**0.00E+00** ± 0.00E+00	1.23E−50 ± 2.86E−50	**0.00E+00** ± 0.00E+00
f_2	30	**1.37E+01** ± 9.86E+00	2.08E+01 ± 1.24E+01	1.83E+01 ± 1.46E+01	1.48E+01 ± 9.57E−01	2.64E+01 ± 1.38E+01	2.01E+01 ± 1.05E+01
f_3	30	2.25E−14 ± 3.07E−14	1.85E−07 ± 2.70E−07	1.09E−14 ± 1.94E−14	3.43E−15 ± 1.59E−15	**5.89E−16** ± 0.00E+00	**5.89E−16** ± 0.00E+00
f_4	30	1.90E−02 ± 8.81E−02	4.37E−09 ± 5.06E−08	1.20E−02 ± 9.14E−02	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00
f_5	30	4.74E−15 ± 3.67E−14	5.62E−07 ± 1.38E−06	4.77E−02 ± 4.08E−01	1.04E−08 ± 2.19E−08	**0.00E+00** ± 0.00E+00	1.40E−13 ± 5.29E−14
f_6	30	3.32E+00 ± 1.18E+01	1.50E−04 ± 6.96E−04	6.27E+00 ± 1.28E+01	**0.00E+00** ± 0.00E+00	2.20E+00 ± 1.96E+00	**0.00E+00** ± 0.00E+00
f_7	30	6.67E−01 ± 6.05E+00	1.93E−03 ± 6.45E−03	2.27E+00 ± 2.45E+01	**0.00E+00** ± 0.00E+00	3.35E+00 ± 1.76E+00	**0.00E+00** ± 0.00E+00
f_8	30	1.82E−01 ± 2.28E+00	1.07E−05 ± 3.77E−05	1.22E+00 ± 5.33E+00	9.59E−13 ± 0.00E+00	**5.89E−16** ± 0.00E+00	**5.89E−16** ± 0.00E+00
f_9	30	2.30E−02 ± 1.01E−01	6.49E−05 ± 2.74E−04	1.38E−02 ± 8.30E−02	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00
f_{10}	30	8.33E+00 ± 2.00E+01	2.99E+00 ± 4.11E+00	8.40E+00 ± 1.37E+01	2.64E−13 ± 2.45E−13	**0.00E+00** ± 0.00E+00	2.69E+01 ± 2.87E-01
f_{11}	30	7.33E+01 ± 1.15E+02	5.48E+01 ± 4.69E+01	7.09E+01 ± 1.20E+02	**0.00E+00** ± 0.00E+00	1.57E+01 ± 1.03E+01	**0.00E+00** ± 0.00E+00
f_{12}	100	7.29E+02 ± 2.46E+02	4.61E−10 ± 6.99E−10	1.63E+04 ± 2.17E+04	3.43E+03 ± 4.82E+02	1.87E−10 ± 5.73E−10	**4.24E−11** ± 3.26E−11
f_{13}	100	3.39E+01 ± 3.38E+01	4.47E+01 ± 3.71E+01	1.01E+02 ± 1.79E+02	3.37E+01 ± 3.63E+00	2.99E+01 ± 1.85E+01	**2.38E+01** ± 1.84E+00
f_{14}	100	1.07E+05 ± 3.13E+06	2.80E+02 ± 2.55E+02	1.95E+09 ± 3.12E+10	4.53E+07 ± 8.41E+06	1.02E+04 ± 6.17E+04	**2.19E+02** ± 9.92E+01
f_{15}	100	3.88E+00 ± 1.42E+01	1.50E−09 ± 1.06E−08	1.19E+02 ± 1.53E+03	2.15E+00 ± 1.73E−01	**7.75 E−10** ± 3.52E−10	2.93E−06 ± 6.61E−06
w/t/l		12/0/3	14/0/1	13/0/2	7/6/2	8/4/3	–

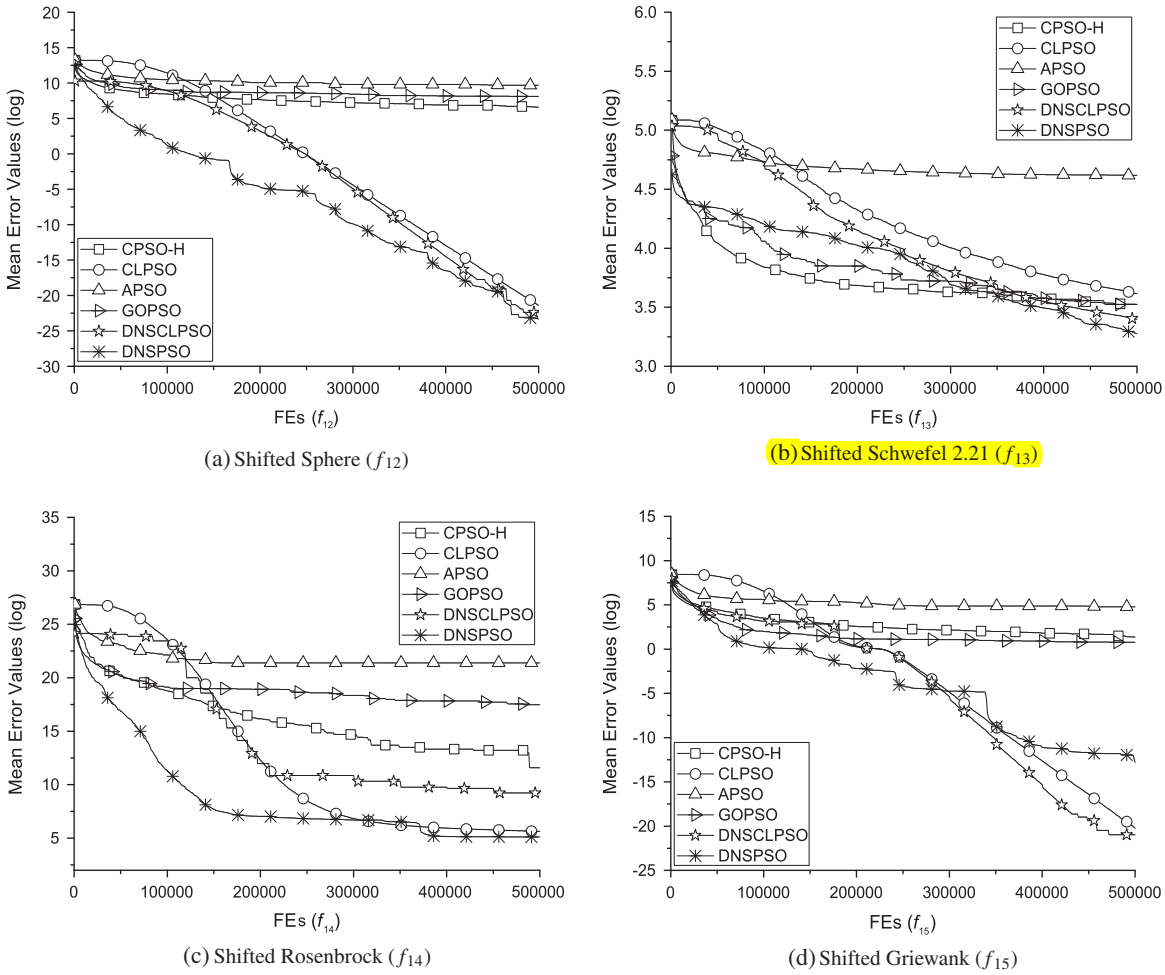


Fig. 4. The convergence curves of CPSO-H, CLPSO, APSO, GOPSO, DNSCLPSO and DNSPSO on high-dimensional functions ($f_{12} - f_{15}$).

Table 7
Average rankings achieved by Friedman test for the six PSO algorithms. The highest ranking is shown in bold.

Algorithms	Rankings
DNSPSO	**4.87**
DNSCLPSO	4.33
GOPSO	4.27
CLPSO	3.07
CPSO-H	2.60
APSO	1.87

Table 8
Wilcoxon test between DNSPSO with other PSO algorithms on functions f_1–f_{15}. The p-values below 0.05 are shown in bold.

DNSPSO vs.	p-Values
CPSO-H	**4.68E−02**
CLPSO	**1.99E−02**
APSO	**1.46E−02**
GOPSO	2.14E−01
DNSCLPSO	1.31E−01

Table 9
Comparison results of mean fitness error values and standard deviations, where "$w/t/l$" means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	BPSO-RVM [36]	CPSO-RVM [36]	CBPSO-RVM [36]	PSO-GM [21]	DNSPSO
Sphere function	3.15E−38 ± 1.77E−37	2.62E−116 ± 1.70E−115	1.65E−146 ± 2.48E−146	5.25E−53 ± 1.52E−52	**0.00E+00** ± 0.00E+00
Quadric function	2.73E+01 ± 2.45E+01	2.84E−04 ± 5.24E−04	3.25E−21 ± 1.16E−20	5.63E−04 ± 9.27E−04	**1.03E−125** ± 5.19E−126
Rosenbrock function	1.82E+01 ± 8.09E+00	1.22E+01 ± 4.29E+01	**2.41E+00** ± 3.46E+00	3.70E+00 ± 4.05E+00	7.06E+00 ± 8.25E+00
Griewank function	9.20E−03 ± 1.10E−02	6.80E−03 ± 1.16E−02	7.20E−03 ± 1.03E−02	3.84E−04 ± 3.20E−03	**0.00E+00** ± 0.00E+00
Rastrigin function	1.17E+01 ± 2.52E+01	2.86E−01 ± 1.88E+01	1.68E−13 ± 1.05E−12	4.56E+00 ± 1.10E+01	**0.00E+00** ± 0.00E+00
Ackley function	1.22E−14 ± 3.38E−15	1.52E−14 ± 2.83E−15	1.17E−14 ± 3.46E−15	8.74E−14 ± 2.16E−15	**5.89E−16** ± 0.00E+00
w*/*t*/*l	6/0/0	6/0/0	5/0/1	5/0/1	–

- BPSO (standard PSO) with RVM (BPSO-RVM) [36].
- CPSO (PSO with constriction factor) and RVM (CPSO-RVM) [36].
- CBPSO (PSO with both constriction factor and inertia weight) with RVM (CBPSO-RVM) [36].
- PSO with Gaussian mutation (PSO-GM) [21].
- Our approach DNSPSO.

For all algorithms, the same settings are used for the common parameters. By the suggestions of [36], the population size (N) and the maximum number of fitness evaluations (MAX_FEs) are set to 20 and 10,000 $*$ 20, respectively (In [36], the maximum number of generations is set to 10,000.).

Table 9 presents the comparison results of mean fitness error values and standard deviations on six benchmark functions. The comparison results among DNSPSO and other four algorithms are summarized as "$w/t/l$" in the last row of the table. As seen, DNSPSO outperforms BPSO-RVM and CPSO-RVM on all test functions. CBPSO-RVM and PSO-GM achieve better results than DNSPSO on Ronsenbrock function, while DNSPSO performs better on the rest five functions.

4.5. Comparison results on CEC 2005 benchmarks

To further verify the performance of DNSPSO, a set of ten CEC 2005 shifted and rotated benchmark functions are used in this section. Table 10 presents simple descriptions of these functions. More detailed definitions of them can be found in [46]. In the experiments, DNSPSO is compared with five other PSO algorithms. The involved algorithms are listed as follows:

- Cooperative PSO (CPSO-H) [1].
- Comprehensive learning PSO (CLPSO) [34].
- Adaptive PSO (APSO) [59].
- PSO with generalized opposition-based learning (GOPSO) [53].
- Combining CLPSO with the proposed diversity mechanism and neighborhood search strategy (DNSCLPSO).
- Our approach DNSPSO.

For the above six algorithms, the same parameter settings are used as described in Section 4.4 except for MAX_FEs. For this test suite, the MAX_FEs is set to 3.00E+05 according to the suggestions of [46]. For each test function, each algorithm is run 25 times. Throughout the experiments, the mean function error values are reported.

Comparison results of mean function error values and standard deviations on the CEC 2005 benchmarks are listed in Table 11, where the best is shown in bold. The comparison results between DNSPSO and other algorithms are summarized as "$w/t/l$" in the last row of the table, which means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors.

Table 10
The ten CEC 2005 benchmark functions used in the experiments, where D is the dimension, and $f(x^o)$ is the global optimum.

Functions	Name	D	$f(x^o)$
f_{cec05_1}	Shifted Sphere Function	30	−450
f_{cec05_2}	Shifted Schwefel's Problem 1.2	30	−450
f_{cec05_3}	Shifted Rotated High Conditioned Elliptic Function	30	−450
f_{cec05_4}	Shifted Schwefel's Problem 1.2 with Noise	30	−450
f_{cec05_5}	Schwefel's Problem 2.6	30	−310
f_{cec05_6}	Shifted Rosenbrock's Function	30	390
f_{cec05_7}	Shifted Rotated Griewank's Function	30	−180
f_{cec05_8}	Shifted Rotated Ackley's Function	30	−140
f_{cec05_9}	Shifted Rastrigin's Function	30	−330
$f_{cec05_{10}}$	Shifted Rotated Rastrigin's Function	30	−330

Table 11
Comparison results of mean function error values and standard deviations on the CEC 2005 benchmarks, where "*w*/*t*/*l*" means that DNSPSO wins in *w* functions, ties in *t* functions and loses in *l* functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	CPSO-H [1]	CLPSO [34]	APSO [59]	GOPSO [53]	DNSCLPSO	DNSPSO
f_{cec05_1}	4.26E−02 ± 1.25E+00	8.01E−13 ± 3.40E−12	7.20E−14 ± 1.38E−13	5.86E−14 ± 6.37E−14	6.53E−14 ± 5.41E−14	**4.29E−14** ± 3.96E−14
f_{cec05_2}	1.34E+03 ± 1.14E+04	3.38E+03 ± 3.19E+03	1.82E−02 ± 9.53E−02	1.26E−04 ± 2.28E−04	2.85E−03 ± 1.52E−03	**2.95E−06** ± 6.21E−06
f_{cec05_3}	1.24E+00 ± 3.64E+01	4.06E−09 ± 1.27E−08	**7.39E−14** ± 1.43E−14	8.31E+06 ± 6.43E+06	6.72E−01 ± 2.30E+00	5.21E+05 ± 4.17E+05
f_{cec05_4}	1.07E+04 ± 7.25E+04	1.12E+04 ± 1.57E+04	9.46E+02 ± 3.30E+03	5.48E+02 ± 4.39E+02	6.26E+02 ± 4.91E+02	**3.79E+00** ± 2.45E+00
f_{cec05_5}	2.74E+04 ± 3.12E+04	1.86E+04 ± 1.26E+04	**2.21E+02** ± 3.27E+02	6.58E+03 ± 5.16E+03	8.41E+03 ± 2.56E+03	2.98E+03 ± 6.24E+03
f_{cec05_6}	1.73E+03 ± 4.23E+04	**1.04E+01** ± 5.67E+01	2.99E+01 ± 1.68E+02	1.53E+01 ± 1.34E+01	1.95E+01 ± 2.03E+01	1.16E+01 ± 8.28E+00
f_{cec05_7}	1.52E+03 ± 1.68E+03	5.42E+03 ± 1.43E+03	5.63E+03 ± 2.85E+03	1.20E+00 ± 1.17E+00	4.28E+01 ± 2.55E+01	**2.96E−02** ± 3.76E−02
f_{cec05_8}	**2.09E+01** ± 3.73E−01	2.10E+01 ± 1.91E−01	2.12E+01 ± 3.57E−01	**2.09E+01** ± 2.39E−01	**2.09E+01** ± 3.04E−01	**2.09E+01** ± 1.83E−01
f_{cec05_9}	3.55E+01 ± 5.88E+01	**1.99E−01** ± 2.59E+00	5.74E+00 ± 9.09E+00	8.36E+01 ± 3.27E+01	3.36E+01 ± 1.69E+01	6.57E+01 ± 2.57E+01
$f_{cec05_{10}}$	2.21E+02 ± 2.87E+02	1.42E+02 ± 8.45E+01	1.33E+02 ± 3.00E+02	2.98E+01 ± 1.43E+01	1.52E−06 ± 2.13E−07	**0.00E+00** ± 0.00E+00
w*/*t*/*l	7/1/2	7/0/3	7/0/3	9/1/0	7/1/2	–

Table 12
Average rankings achieved by Friedman test for the six PSO algorithms on the CEC 2005 benchmarks. The highest ranking is shown in bold.

Algorithms	Rankings
DNSPSO	**4.85**
DNSCLPSO	3.95
GOPSO	3.85
APSO	3.30
CLPSO	2.90
CPSO-H	2.15

From the results of Table 11, DNSPSO achieves better results than CPSO-H on 7 functions, while CPSO-H performs better than DNSPSO on 2 functions. Compared to CLPSO and APSO, DNSPSO outperforms them on 7 functions, while CLPSO and APSO achieves better results on three functions. DNSPSO surpasses GOPSO on all functions except for f_{cec05_8}. On this function, both of them fall into the same local optimum.

By hybridization of CLPSO and our proposed strategies, DNSCLPSO performs better than CLPSO on 7 functions. It demonstrates that our proposals are helpful to improve the performance of PSO variants.

Table 12 shows the average ranking of CPSO-H, CLPSO, APSO, GOPSO, DNSCLPSO and DNSPSO. The highest ranking is shown in bold. As seen, the performance of the five algorithms can be sorted by average ranking into the following order: DNSPSO, DNSCLPSO, GOPSO, APSO, CLPSO, and CPSO-H. The highest average ranking is obtained by the DNSPSO algorithm. It demonstrates that DNSPSO is the best one among the six PSO algorithms for the CEC 2005 benchmarks.

Table 13
The CEC 2010 large-scale benchmark functions used in the experiments, where D is the dimension, and $f(x^o)$ is the global optimum.

Functions	Name	D	$f(x^o)$
F_1	Shifted Elliptic Function	1000	0
F_2	Shifted Rastrigin's Function	1000	0
F_3	Shifted Ackley's Function	1000	0
F_4	Single-group Shifted and m-rotated Elliptic Function	1000	0
F_5	Single-group Shifted and m-rotated Rastrigin's Function	1000	0
F_6	Single-group Shifted and m-rotated Ackley's Function	1000	0
F_7	Single-group Shifted m-dimensional Schwefel's Problem 1.2	1000	0
F_8	Single-group Shifted m-dimensional Rosenbrock's Function	1000	0
F_9	$\frac{D}{2m}$-Group Shifted and m-rotated Elliptic Function	1000	0
F_{10}	$\frac{D}{2m}$-Group Shifted and m-rotated Rastrigin's Function	1000	0
F_{11}	$\frac{D}{2m}$-Group Shifted and m-rotated Ackley's Function	1000	0
F_{12}	$\frac{D}{2m}$-Group Shifted m-dimensional Schwefel's Problem 1.2	1000	0
F_{13}	$\frac{D}{2m}$-Group Shifted m-dimensional Rosenbrock's Function	1000	0
F_{14}	$\frac{D}{m}$-Group Shifted and m-rotated Elliptic Function	1000	0
F_{15}	$\frac{D}{m}$-Group Shifted and m-rotated Rastrigin's Function	1000	0
F_{16}	$\frac{D}{m}$-Group Shifted and m-rotated Ackley's Function	1000	0
F_{17}	$\frac{D}{m}$-Group Shifted m-dimensional Schwefel's Problem 1.2	1000	0
F_{18}	$\frac{D}{m}$-Group Shifted m-dimensional Rosenbrock's Function	1000	0
F_{19}	Shifted Schwefel's Problem 1.2	1000	0
F_{20}	Shifted Rosenbrock's Function	1000	0

4.6. Comparison results on CEC 2010 large-scale benchmarks

In Section 4.1, the high-dimensional problems (f_{12}-f_{14}) are separable. To test more complex and nonseparable problems, the CEC 2010 large-scale benchmarks are used in this section. Table 13 presents simple descriptions of these functions. More detailed definitions of them can be found in [49].

In the experiments, DNSPSO is compared with CLPSO, GOPSO, DNSCLPSO and the competition winner of CEC 2010 Special Session and Competition on Large Scale Global Optimization (MA-SW-Chains) [39]. MA-SW-Chains is a memetic algorithm, which assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. Here, we do not compare DNSPSO with APSO and CPSO-H on this test suite, because these two algorithms are worse than CLPSO and GOPSO.

The parameter settings of these five algorithms are listed as follows. For CLPSO, GOPSO, DNSCLPSO and DNSPSO, the population size is set to 100. For the rest parameters, the same settings are used as described in Section 4.4. For MA-SW-Chains, the same parameter settings are used as [39]. By the suggestions of [49], the maximum number of fitness evaluations (MAX_FEs) is set to 3.00E+06 for all algorithms. Each algorithm is run 25 times for a function. Throughout the experiments, the mean function values and standard deviations are reported.

Comparison results of mean function values and standard deviations on the CEC 2010 large-scale benchmarks are presented in Table 14, where the best and the second best results are highlighted in boldface and italic, respectively. The comparison results between DNSPSO and other algorithms are summarized as "$w/t/l$" in the last row of the table, which means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors.

From the results of Table 14, DNSPSO outperforms CLPSO on 19 functions, while CLPSO only achieves better results on f_5. DNSPSO surpasses GOPSO on all test functions. By combining CLPSO with our proposed strategies, DNSCLPSO achieves significant improvements on most functions. DNSCLPSO outperforms DNSPSO on seven functions, while DNSPSO obtains better results on the rest 13 functions. However, DNSPSO and other three PSO variants are not suitable for large-scale optimization

Table 14
Comparison results of mean function values and standard deviations on the CEC 2010 large-scale benchmarks, where "$w/t/l$" means that DNSPSO wins in w functions, ties in t functions and loses in l functions, compared with its competitors.

Functions	CLPSO [34]	GOPSO [53]	MA-SW-Chains [39]	DNSCLPSO	DNSPSO
F_1	7.80E+08 ± 6.79E+07	1.13E+10 ± 2.37E+09	**2.10E−14** ± 1.99E−14	1.67E+08 ± 7.30E+06	*1.87E+07* ± 1.73E+06
F_2	6.06E+03 ± 1.40E+02	9.25E+03 ± 5.62E+03	**8.10E+02** ± 5.88E+01	*5.66E+03* ± 8.29E+01	5.85E+03 ± 4.23E+02
F_3	2.04E+01 ± 8.29E−02	1.97E+01 ± 7.21E−02	**7.28 E−13** ± 3.43E−13	*1.91E+01* ± 7.89E−01	1.93E+01 ± 4.44E−02
F_4	9.98E+13 ± 9.44E+12	1.92E+13 ± 1.04E+12	**3.53E+11** ± 3.12E+10	8.99E+12 ± 1.22E+12	*2.25E+12* ± 1.96E+11
F_5	*1.44E+08* ± 2.63E+07	1.62E+08 ± 1.46E+08	1.68E+08 ± 1.04E+08	**8.52E+07** ± 1.43E+07	1.57E+08 ± 2.32E+07
F_6	6.68E+06 ± 7.99E+05	1.10E+07 ± 2.33E+07	*8.14E+04* ± 2.84E+05	**2.41E+01** ± 9.95E−01	1.75E+06 ± 2.75E+05
F_7	9.03E+09 ± 8.80E+08	1.41E+09 ± 2.62E+09	**1.03E+02** ± 8.70E+01	6.57E+08 ± 2.44E+08	*8.60E+06* ± 2.55E+05
F_8	1.11E+08 ± 1.40E+07	9.68E+07 ± 1.24E+10	*1.41E+07* ± 3.68E+07	9.19E+07 ± 3.50E+07	**1.31E+07** ± 4.65E+06
F_9	8.96E+09 ± 1.29E+08	4.28E+10 ± 2.73E+10	**1.41E+07** ± 1.15E+06	1.57E+09 ± 6.43E+07	*3.16E+08* ± 2.49E+06
F_{10}	1.24E+04 ± 2.80E+02	9.69E+03 ± 1.28E+04	**2.07E+03** ± 1.44E+02	7.63E+03 ± 2.57E+01	*6.90E+03* ± 1.88E+02
F_{11}	2.28E+02 ± 1.03E−01	2.01E+02 ± 1.35E+02	**3.80E+01** ± 7.35E+00	*1.68E+02* ± 9.13E+00	1.76E+02 ± 5.74E+00
F_{12}	7.26E+06 ± 1.08E+06	5.91E+06 ± 2.61E+06	**3.62E−06** ± 5.92E−07	1.00E+06 ± 5.83E+03	*2.34E+05* ± 1.81E+04
F_{13}	6.73E+08 ± 1.55E+08	5.23E+09 ± 3.28E+09	**1.25E+03** ± 5.72E+02	2.20E+08 ± 3.41E+07	*1.15E+06* ± 1.81E+05
F_{14}	2.09E+10 ± 7.32E+08	5.48E+10 ± 4.16E+10	**3.11E+07** ± 1.93E+06	*3.15E+09* ± 8.62E+07	3.81E+09 ± 7.15E+07
F_{15}	1.47E+04 ± 2.49E+02	1.24E+04 ± 9.40E+03	**2.74E+03** ± 1.22E+02	8.26E+03 ± 8.78E+01	*7.52E+03* ± 4.34E+01
F_{16}	4.24E+02 ± 2.85E−01	3.97E+02 ± 2.58E+02	**9.98E+01** ± 1.40E+01	4.04E+02 ± 9.27E−01	*3.86E+02* ± 4.15E−01
F_{17}	1.15E+07 ± 1.04E+06	1.05E+07 ± 8.18E+06	**1.24E+00** ± 1.25E−01	1.56E+06 ± 2.41E+04	*6.13E+05* ± 6.03E+03
F_{18}	5.69E+10 ± 8.51E+09	5.46E+11 ± 7.19E+10	**1.30E+03** ± 4.36E+02	2.81E+10 ± 3.05E+09	*2.29E+07* ± 9.39E+06
F_{19}	3.82E+07 ± 8.43E+06	8.78E+06 ± 5.42E+06	**2.85E+05** ± 1.78E+04	*5.05E+06* ± 7.09E+04	5.33E+06 ± 1.66E+06
F_{20}	8.44E+10 ± 2.18E+09	2.22E+11 ± 3.15E+10	**1.07E+03** ± 7.29E+01	4.23E+10 ± 3.61E+09	*2.02E+08* ± 1.92E+08
w*/*t*/*l	19/0/1	20/0/0	2/0/18	13/0/7	–

Table 15
Average rankings achieved by Friedman test for the CEC 2010 large-scale benchmarks. The highest ranking is shown in bold.

Algorithms	Rankings
MA-SW-Chains	**4.70**
DNSPSO	3.70
DNSCLPSO	3.40
GOPSO	1.65
CLPSO	1.55

problems. Compared to the competition winner of CEC 2010 Special Session on Large-Scale Global Optimization (MA-SW-Chains) [39], DNSPSO performs better than MA-SW-Chains on only two functions.

Table 15 shows the average ranking of CLPSO, GOPSO, MA-SW-Chains, DNSCLPSO and DNSPSO. The highest ranking is shown in bold. As seen, the performance of the five algorithms can be sorted by average ranking into the following order: MA-SW-Chains, DNSPSO, DNSCLPSO, GOPSO, and CLPSO. The best and the second best algorithms are MA-SW-Chains and DNSPSO, respectively.

5. Conclusions

This paper presents an enhanced PSO algorithm called DNSPSO to solve complex optimization problems. The proposed approach employs two strategies including a diversity enhancing mechanism and two neighborhood search strategies. The former strategy is helpful to increase the swarm diversity by adjusting the dissimilarities among particles. The latter strategy is beneficial for accelerating the convergence rate because of the attraction of the previous best particles and the global best particle. By combining these two strategies, DNSPSO achieves a trade-off between the exploration and exploitation abilities. To verify the performance of DNSPSO, different types of benchmark functions are tested in the experiments.

The values of the parameters p_r and p_{ns} affect the performance of DNSPSO. Simulation results show that fixed p_r and p_{ns} are not suitable for all functions. A smaller p_r works well on low-dimensional functions, while a larger p_r is suitable for high-dimensional functions. The values of p_{ns} does not seriously affect the performance of DNSPSO and $p_{ns} \in (0,1)$ can achieve good results.

PSO based on diversity enhancing mechanism (DPSO) or on neighborhood search (NSPSO) cannot achieve promising results. By combining them, DNSPSO obtains an excellent performance. The results of mean computational time shows that our proposed algorithm does not increase computational time compared with the standard PSO under the same maximum number of fitness evaluations.

For the low-dimensional problems including two kinds of benchmarks, DNSPSO achieves better results than CPSO-H, CLPSO, APSO, GOPSO and PSO with RVM. By integrating our proposed strategies into CLPSO, DNSCLPSO obtains significantly improvements. For high-dimensional problems, though DNSPSO outperforms CLPSO, GOPSO and DNSCLPSO, all these PSO variants are not good choices for the CEC 2010 large-scale benchmarks.

The parameter k (the neighborhood radius) may affect the effectiveness of the neighborhood search. How to tune k will be investigated in the future work.

Acknowledgments

The authors thank the editor and anonymous reviewers for their detailed and constructive comments that help us to increase the quality of this work. This work is supported by the Science and Technology Plan Projects of Jiangxi Provincial Education Department (Nos. GJJ12641, GJJ12633, GJJ12307), the Youth Foundation of Nanchang Institute of Technology (No. 2012KJ021), and the National Natural Science Foundation of China (Nos. 61070008, 61165004, 61261039).

References

[1] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[2] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.
[3] X.J. Cai, Z.H. Cui, J.C. Zeng, Y. Tan, Performance-dependent adaptive particle swarm optimization, International Journal of Innovative Computing, Information and Control 3 (6B) (2007) 1697–1706.
[4] A. Cervantes, I.M. Galván, P. Isasi, AMPSO: a new particle swarm method for nearest neighborhood classification, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 39 (5) (2009) 1082–1091.
[5] P. Chakraborty, S. Das, G.G. Roy, A. Abraham, On convergence of the multi-objective particle swarm optimizers, Information Sciences 181 (8) (2011) 1411–1425.
[6] Y.P. Chen, W.C. Peng, M.C. Jian, Particle swarm optimization with recombination and dynamic linkage discovery, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 37 (6) (2007) 1460–1470.
[7] S.C. Chu, P.W. Tsai, Computational intelligence based on behaviors of cats, international journal of innovative computing, International Journal of Innovative Computing, Information and Control 3 (1) (2007) 163–173.
[8] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.
[9] Z.H. Cui, X.J. Cai, J.C. Zeng, Chaotic performance-dependent particle swarm optimization, International Journal of Innovative Computing, Information and Control 5 (4) (2009) 951–960.
[10] Z.H. Cui, X.J. Cai, J.C. Zeng, Y. Yin, PID-controlled particle swarm optimization, Journal of Multiple-Valued Logic and Soft Computing 16 (6) (2010) 585–610.
[11] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.
[12] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[13] J. Derrac, S. GarcÍa, 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 and Evolutionary Computation 1 (1) (2011) 3–18.
[14] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 26 (1996) 29–41.
[15] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.

[16] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach, Information Sciences 216 (2012) 50–92.
[17] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (20) (2010) 2044–2064.
[18] S. García, F. Herrera, An extension on Statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[19] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, Journal Heuristics 15 (2009) 617–644.
[20] S. Ghosh, S. Das, D. Kundu, K. Suresh, A. Abraham, Inter-particle communication and search-dynamics of lbest particle swarm optimizers: an analysis, Information Sciences 182 (1) (2012) 156–168.
[21] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
[22] S.Y. Ho, H.S. Lin, W.H. Liauh, S.H. Ho, OPSO: orthogonal particle swarm optimization and its application to task assignment problems, IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans 38 (2) (2008) 288–298.
[23] S. Hsieh, T. Sun, C. Liu, S. Tsai, Efficient population utilization strategy for particle swarm optimizer, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 39 (2) (2009) 444–456.
[24] J. Hu, J. Zeng, Y. Tan, A diversity-guided particle swarm optimizer for dynamic environments, in: Proceedings of Life System Modeling and Simulation, 2007, pp. 239–247.
[25] X. Hu, R.C. Eberhart, Multiobjective optimization using dynamic neighborhood particle swarm optimization, in: Proceedings of the Congress Evoluationary Computer, 2002, pp. 1677–1681.
[26] C.H. Hsul, W.J. Shyr, K.H. Kuo, Optimizing multiple interference cancellations of linear phase array based on particle swarm optimization, Journal of Information Hiding and Multimedia Signal Processing 1 (4) (2010) 292–300.
[27] D. Karaboga, An Idea Based on Honey BEE Swarm for Numerical Optimization, Technical report TR06, Computer Engineering Department, Erciyes University, Turkey, 2005.
[28] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1391–1938.
[29] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[30] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of IEEE Congress on Evolutionary Computation, 2002, 1671–1676.
[31] C. Li, S. Yang, An adaptive learning particle swarm optimizer for function optimization, in: Proceedings of the Congress Evoluationary Computer, 2009, pp. 381–388.
[32] C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 42 (3) (2012) 627–646.
[33] S. Li, M.T. Tan, I.W. Tsang, J.T. Kwok, A hybrid PSO-BFGS strategy for global optimization of multimodal functions, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 41 (4) (2011) 1003–1014.
[34] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (2006) 281–295.
[35] H. Liu, A. Abraham, An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems, Journal of Universal Computer Science 13 (9) (2007) 1309–1331.
[36] H. Lu, P. Sriyanyong, Y.H. Song, T. Dillon, Experimental study of a new hybrid PSO with mutation for economic dispatch with non-smooth cost function, International Journal of Electrical Power and Energy 32 (9) (2010) 921–935.
[37] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 204–210.
[38] A.S. Mohais, R. Mendes, C. Ward, C. Posthoff, Neighborhood re-structuring in particle swarm optimization, in: Proceedings of Australian Conference on Artificial Intelligence, 2005, pp. 776–785.
[39] D. Molina, M. Lozano, F. Herrera, MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization, in: Proceedings of IEEE Congress Evolutionary Computation, 2010, pp. 3153–3160.
[40] M. Pant, T. Radha, V.P. Singh. A simple diversity guided particle swarm optimization, in: Proceedings of IEEE Congress Evolutionary Computation, 2007, pp. 3294–3299.
[41] T. Peram, K. Veeramachaneni, C.K. Mchan, Fitness-distance-ratio based particle swarm optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2003, pp. 174–181.
[42] J. Riget, J.S. Vesterstom, Adiversity-Guided Particle Swarm Optimizer – the arPSO. Technical report, EVAlife, Denmark, 2002.
[43] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the Congress Evoluationary Computer, 1998, pp. 69–73.
[44] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[45] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1958–1962.
[46] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical report, Nanyang Technological University, 2005.
[47] J. Sun, B.W. Xu, W. Fang, A diversity-guided quantum-behaved particle swarm optimization algorithm, in: International Conference on Simulated Evolution and Learning, 2006, pp. 497–504.
[48] C. Sun, J. Zeng, J. Pan, An improved vector particle swarm optimization for constrained optimization problems, Information Sciences 181 (6) (2011) 1153–1163.
[49] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization, Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2010.
[50] K. Tang, X. Yao, P.N. Suganthan, C. Macnish, Y. Chen, C. Chen, Z. Yang, Benchmark Functions for the CEC'2008 Special Session and Competition on High-Dimensional Real-Parameter Optimization, Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.
[51] R.K. Ursem, Diversity-guided evolutionary algorithms, in: Proceedings of the Parallel Problem from Nature, vol. VII, 2002, pp. 462–471.
[52] H. Wang, Z. Wu, S. Rahnamayan, C. Li, S. Zeng, D. Jiang, Particle swarm optimization with simple and efficient neighbourhood search strategies, International Journal of Innovative Computing and Applications 3 (2) (2011) 97–104.
[53] H. Wang, Z.J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, Information Sciences 181 (20) (2011) 4699–4714.
[54] H. Wang, Z. Wu, S. Zeng, D. Jiang, Y. Liu, J. Wang, X. Yang, A simple and fast particle swarm optimization, Journal of Multiple-Valued Logic and Soft Computing 16 (6) (2010) 611–629.
[55] W. Wang, H. Wang, S. Rahnamayan, Improving comprehensive learning particle Swarm optimizer using generalized opposition-based learning, International Journal of Modelling, Identification and Control 14 (4) (2011) 310–316.
[56] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences 180 (20) (2011) 4515–4538.
[57] Z. Yang, K. Tang, X. Yao, Large scale evolutionary using cooperative coevolution, Information Sciences 178 (2008) 2985–2999.

[58] F. Yano, T. Shohdohji, Y. Toyoda, An improvement of particle swarm optimization with a neighborhood search algorithm, Industrial Engineering and Management Systems 6 (1) (2007) 64–71.
[59] Z. Zhan, J. Zhang, Y. Li, H. Chung, Adaptive particle swarm optimization, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 39 (6) (2009) 1362–1381.
[60] W.J. Zhang, X.F. Xie, DEPSO: hybrid particle swarm with differential evolution operator, Proceedings of IEEE International Conference on Systems, Man and Cybernetics (2003) 3816–3821.