Applied Mathematics and Computation 242 (2014) 265–276

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Enhanced comprehensive learning particle swarm optimization

Xiang Yu, Xueqing Zhang [*]

Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Keywords:
Global optimization
Particle swarm optimization
Comprehensive learning

ABSTRACT

Comprehensive learning particle swarm optimization (CLPSO) is a state-of-the-art metaheuristic that encourages a particle to learn from different exemplars on different dimensions. It is able to locate the global optimum region for many complex multimodal problems as it is excellent in preserving the particles' diversity and thus preventing premature convergence. However, CLPSO has been noted for low solution accuracy. This paper proposes two enhancements to CLPSO. First, a perturbation term is added into each particle's velocity update procedure to achieve high performance exploitation. Normative knowledge about dimensional bounds of personal best positions is used to appropriately activate the perturbation based exploitation. Second, the particles' learning probabilities are determined adaptively based on not only rankings of personal best fitness values but also the particles' exploitation progress to facilitate convergence. Experiments conducted on various benchmark functions demonstrate that the two enhancements successfully overcome the low solution accuracy weakness of CLPSO.

1. Introduction

Introduced in 1995 [1], particle swarm optimization (PSO) is a modern metaheuristic that has been applied to solve real-world optimization problems in a wide range of areas. PSO simulates the movements of organisms in a bird flock or fish school. PSO is population-based and finds the optimum using a swarm of particles, with each particle representing a candidate solution. Compared with traditional optimization methods such as linear programming, nonlinear programming, and dynamic programming, PSO does not require the objective and constraints of the optimization problem to be continuous, differentiable, linear, or convex, and PSO usually can efficiently solve large-scale problems. PSO shares many similarities with evolutionary computation based metaheuristics such as genetic algorithm and differential evolution, but PSO basically does not use any evolution operator (e.g. crossover, mutation, or selection), thus PSO is simpler in concept and easier to implement.

In PSO, all the particles "fly" in the search space. Each particle, denoted as i, is associated with a position, a velocity, and a fitness that indicates its performance. PSO relies on iterative learning to find the optimum. In each iteration (or generation), i adjusts its velocity according to its previous velocity, its historical best position (i.e. personal best position), and also the personal best positions of its neighborhood particles. There are many PSO variants that differ in how to use the neighborhood search experiences for the velocity update. In global PSO (GPSO) [2], the historical best position out of the entire swarm (i.e. global best position) is used to update i's velocity. GPSO does not perform well on complex problems, thus various local PSOs (LPSOs) have been studied [3]. In a LPSO, a static topological structure (e.g. ring, pyramid, or von Neumann) is constructed

[*] Corresponding author.
E-mail addresses: xyuac@ust.hk (X. Yu), zhangxq@ust.hk (X. Zhang).

http://dx.doi.org/10.1016/j.amc.2014.05.044
0096-3003/© 2014 Elsevier Inc. All rights reserved.

and the neighborhood of i consists of i itself and particles that i directly connects to. Instead of referring to the global best position, the historical best position of the neighborhood (i.e. local best position) is used for updating i's velocity. Compared with LPSO, standard PSO (SPSO) [4] maintains a dynamic topological structure. Fully informed PSO (FIPSO) [5] uses a weighted average of the personal best positions of all the particles in i's neighborhood to update i's velocity. For GPSO, LPSO, SPSO, and FIPSO, once the neighborhood related exemplar position has been determined, it is used to update a particle's velocity on all dimensions.

Recently, recognizing the fact that one exemplar does not always offer a good guide on every dimension, two PSO variants, namely comprehensive learning PSO (CLPSO) [6] and orthogonal learning PSO (OLPSO) [7], have been proposed in literature to encourage a particle to learn from different exemplars on different dimensions. In CLPSO, for each particle and on each dimension, a learning probability controls whether to learn from the personal best position of the particle itself or that of some other randomly selected particle. OLPSO adopts orthogonal experimental design to determine the best combination of learning from a particle's personal best position or its neighborhood's historical best position (i.e. global/local best position) on different dimensions with a polynomial number of experimental samples. OLPSO has two versions, global version OLPSO-G and local version OLPSO-L. Zhan et al. [7] compared OLPSO-G, OLPSO-L, CLPSO, and other PSO variants on a number of benchmark functions. The results showed that OLPSO-L and CLPSO significantly outperform OLPSO-G and traditional PSOs (including GPSO, LPSO, SPSO, and FIPSO) on many complex multimodal problems because they are able to preserve the swarm's diversity and thus locate the global optimum region. CLPSO performs worse than traditional PSOs and OLPSO-G on unimodal and simple multimodal problems and OLPSO-L on complex multimodal problems as it is lower in solution accuracy.

Some researchers have worked on improving the performance of CLPSO on single-objective global optimization problems and extending CLPSO to multiobjective optimization. Liang and Suganthan [8] proposed an adaptive CLPSO with history learning, wherein the particles' learning probabilities are adjusted adaptively based on the value that has achieved the biggest improvement in past several generations and the particle velocity update takes into account the historical improving direction. Zheng et al. [9] introduced a mechanism to CLPSO that adaptively sets the values of the inertia weight and acceleration coefficient based on evolutionary information of the particles. In [10], through combining chaotic local search with CLPSO, a memetic scheme enables stagnant particles that cannot be improved by the comprehensive learning strategy to escape from local optima and enables some elite particles to do fine-grained local searches around promising regions. The Pareto dominance concept was integrated with CLPSO to handle multiobjective optimization problems in [11–13].

This paper proposes two novel enhancements to CLPSO to improve the algorithm performance in terms of solution accuracy. The enhanced method is called enhanced CLPSO (ECLPSO). The two enhancements are respectively perturbation based exploitation and adaptive learning probabilities. Specifically, a perturbation term is added into each particle's velocity update procedure to achieve high performance exploitation, with normative knowledge about dimensional bounds of the personal best positions used to appropriately activate the perturbation based exploitation; and the particles' learning probabilities are adaptively determined based on not only rankings of the personal best fitness values but also the particles' exploitation progress to facilitate convergence.

The rest of this paper is organized as follows. In Section 2, CLPSO and research trends on PSO are reviewed. Section 3 elaborates the rationales and implementation details of the two enhancements made in ECLPSO and the algorithm framework of ECLPSO. In Section 4, the performance of ECLPSO is evaluated on fourteen typical unimodal, multimodal, and rotated benchmark functions. Section 5 concludes the paper.

2. Background

2.1. Comprehensive learning particle swarm optimization

Let there be D decision variables, the swarm of particles move in a D-dimensional space. Each particle i is associated with a D-dimensional position $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,D})$ and a D-dimensional velocity $V_i = (V_{i,1}, V_{i,2}, \ldots, V_{i,D})$. In each generation, V_i and X_i on each dimension are updated as described in (1) and (2).

$$V_{i,d} = wV_{i,d} + cr_d(E_{i,d} - X_{i,d}) \tag{1}$$

$$X_{i,d} = X_{i,d} + V_{i,d} \tag{2}$$

where d is the dimension index; w is the inertia weight; $E_i = (E_{i,1}, E_{i,2}, \ldots, E_{i,D})$ is the guidance vector of exemplars; c is the acceleration coefficient and usually $c = 1.5$ [6,7]; and r_d is a random number in [0, 1]. $V_{i,d}$ is usually clamped to a pre-specified positive value $V_d^{\max}$. If $V_{i,d} > V_d^{\max}$, then $V_{i,d}$ is set to $V_d^{\max}$; or if $V_{i,d} < -V_d^{\max}$, then $V_{i,d}$ is set to $-V_d^{\max}$. Let $\underline{X_d}$ and $\overline{X_d}$, respectively be the lower and upper bounds of the search space on the dth dimension, $V_d^{\max}$ is usually set as 20% of $\overline{X_d} - \underline{X_d}$ [7]. The personal best position of particle i is denoted as $P_i = (P_{i,1}, P_{i,2}, \ldots, P_{i,D})$. After X_i is updated, X_i is evaluated and will replace P_i if it has a better fitness value.

PSO possesses two important characteristics: exploration and exploitation. Exploration is the ability to search different regions for locating a good solution, while exploitation is the ability to concentrate the search around a small region for refining a hopeful solution. The inertia weight w linearly decreases during the run of CLPSO in order to balance exploration and exploitation. Specifically, let $k_{\max}$ be the maximum number of generations specified, in each generation k, w is updated according to (3).

$$w = w_{\max} - \frac{k}{k_{\max}}(w_{\max} - w_{\min}) \tag{3}$$

where $w_{\max}$ and $w_{\min}$ are respectively the maximum and minimum inertia weights. Usually $w_{\max} = 0.9$ and $w_{\min} = 0.4$ [6,7].

The exemplar $E_{i,d}$ can be the dth dimension of P_i or that of the personal best position of some other particle j which is selected from a tournament procedure. The decision to learn whether from $P_{i,d}$ or $P_{j,d}$ depends on a learning probability L_i. On each dimension, a random number is generated. If the number is no less than L_i, the corresponding dimension will learn from P_i; otherwise, it will learn from P_j. To select j, first two particles are randomly chosen excluding i; then, the two particles' personal best fitness values are compared; finally, the winner's personal best position is used as the exemplar. If all the exemplars of a particle are its own personal best position, CLPSO will randomly choose one dimension to learn from some other particle's personal best position.

The learning probabilities are set such that each particle has a different learning probability, thus the particles have different levels of exploration and exploitation capabilities. An empirical expression (4) is developed in CLPSO to set the L_i value for each particle i.

$$L_i = L_{\min} + (L_{\max} - L_{\min})\frac{\exp\left(\frac{10(i-1)}{N-1}\right) - 1}{\exp(10) - 1} \tag{4}$$

where $L_{\max}$ and $L_{\min}$ are respectively the maximum and minimum learning probabilities, usually $L_{\max} = 0.5$ and $L_{\min} = 0.05$ [6]; and N is the number of particles in the swarm.

To ensure that a particle learns form good exemplars and to minimize time wasted on poor directions, CLPSO allows the particle to learn from the same exemplars until its fitness value ceases improving for a certain consecutive number of generations called refreshing gap g, then the exemplars are re-determined. Usually $g = 7$ [6,7].

CLPSO computes the fitness value of a particle and updates the particle's personal best position only if the particle is feasible (i.e. within the search space). If a particle is infeasible, since all the exemplars are feasible, the particle will eventually be drawn back to the search space.

2.2. Research trends on particle swarm optimization

PSO has attracted extensive research interests. There are two general trends about improving the performance of PSO: one is to develop better topological structures that exploit useful information in the particles' search experiences, and the other is to hybridize PSO with other search techniques.

The PSO variants such as GPSO, LPSO, SPSO, FIPSO, CLPSO, and OLPSO are examples of the first trend. Chen et al. [14] combined GPSO with an aging leader and challengers to overcome premature convergence. The median positions of particles and the worst and median fitness values of the swarm were exploited in [15].

Operators such as crossover, mutation, and selection in evolutionary computation based metaheuristics have been integrated with PSO to increase the swarm's diversity [16], help avoid premature convergence [17–19], and preserve the best particles [20]. Deflection, stretching, and repulsion were used in [21] to help preventing particles from moving close to a previously discovered local optimum. A dissipative PSO that persistently injects randomness into the swarm through reinitialization of particles was studied in [22]. Van den Bergh and Engelbrecht [23] proposed a cooperative PSO that uses one-dimensional swarms to search on each dimension independently and consolidate the search results by a global swarm. Al-kazemi and Mohan [24] partitioned the swarm of particles into subswarms, with each subswarm behaving differently and information being exchanged among the subswarms. Feng et al. [25] updated the inertia weight according to chaotic dynamics. Zhan et al. [26] and Hu et al. [27] studied adjusting algorithm parameters adaptively. Zhao et al. [28] incorporated principal component analysis and line search into PSO. Stochastic coefficients were replaced with random dimension selection in [29]. Sun et al. [30] presented quantum-behaved PSO with Gaussian distributed local attractors. Particle migration was introduced in [31]. Deb and Padhye [32] established an algorithmic linking between PSO and genetic algorithm.

3. Enhanced comprehensive learning particle swarm optimization

3.1. Perturbation based exploitation

PSO algorithms generally conduct exploration in the beginning of a run to find a possible region that contains the global optimum and then exploit the particular small region. CLPSO is strong in exploration, but it is poor at exploitation or local search and thus fails to improve solution accuracy [7]. To achieve effective and efficient exploitation, three issues need to be addressed: (1) When to perform exploitation? (2) How to delineate the small region that exploitation is to concentrate on? and (3) How to select an appropriate search granularity for exploitation?

ECLPSO implements high performance exploitation through using normative knowledge. The normative knowledge, as defined in [33], is dimensional intervals relevant to all the personal best positions. The normative knowledge has a structure

Table 1
Structure of the normative knowledge.

Dimension	1	2	...	D
Present dimensional lower bound	$\underline{P_1}$	$\underline{P_2}$	...	$\underline{P_D}$
Present dimensional upper bound	$\overline{P_1}$	$\overline{P_2}$	...	$\overline{P_D}$

shown in Table 1, where $\underline{P_d}$ and $\overline{P_d}$ are respectively the lower and upper bounds relevant to all the personal best positions on the dth dimension, i.e. $\underline{P_d} = \text{Min}\{P_{1,d}, P_{2,d}, \ldots, P_{N,d}\}$ and $\overline{P_d} = \text{Max}\{P_{1,d}, P_{2,d}, \ldots, P_{N,d}\}$.

ECLPSO performs local searches on each dimension independently. On each dimension d, if (5) is true, then all the particles' historical best experiences crowd in the small interval $[\underline{P_d}, \overline{P_d}]$ on that specific dimension, and a local search can thus be activated on the dth dimension with the purpose of finding the global optimum X_d^* around $[\underline{P_d}, \overline{P_d}]$ on that dimension.

$$\overline{P_d} - \underline{P_d} \leqslant \alpha(\overline{X_d} - \underline{X_d}) \text{ and } \overline{P_d} - \underline{P_d} \leqslant \beta \tag{5}$$

where α is the relative ratio and α is a small positive number that is much less than 1; and β is the small absolute bound. As the search space $[\underline{X_d}, \overline{X_d}]$ can be very large, the combined use of the relative ratio and absolute bound ensures that the normative interval $[\underline{P_d}, \overline{P_d}]$ is indeed small enough. A small normative interval usually indicates that the particles have reached an equilibrium state and the interval hopefully contains the global optimum on the corresponding dimension. Empirical values chosen for α and β are 0.01 and 2, respectively.

Now that the timing and region issues for exploitation have been solved, the last search granularity issue can be settled through adding a perturbation term into the particle velocity update procedure. Eq. (1) is modified to become (6) in the case of local search.

$$V_{i,d} = w_{\text{PbE}} V_{i,d} + c r_d \left(E_{i,d} + \eta \left(\frac{\underline{P_d} + \overline{P_d}}{2} - E_{i,d} \right) - X_{i,d} \right) \tag{6}$$

where w_{PbE} is the inertia weight used exclusively for exploitation; and η is the perturbation coefficient. As can be seen from (6), each particle i is pulled towards $E_{i,d}$ plus a perturbation term $\eta\left(\frac{\underline{P_d}+\overline{P_d}}{2} - E_{i,d}\right)$ on the dth dimension. The perturbation term takes this form based on the following considerations: first, it is obvious that $\frac{\underline{P_d}-\overline{P_d}}{2} \leqslant \frac{\underline{P_d}+\overline{P_d}}{2} - E_{i,d} \leqslant \frac{\overline{P_d}-\underline{P_d}}{2}$, hence the perturbation term is relative to the normative interval's size $\overline{P_d} - \underline{P_d}$, and the smaller the interval size, the smaller the perturbation; and second, the global optimum on the dth dimension, i.e. X_d^*, often is located near the normative interval's center $\frac{\underline{P_d}+\overline{P_d}}{2}$, hence the perturbation term also is relative to the distance between $E_{i,d}$ and $\frac{\underline{P_d}+\overline{P_d}}{2}$, and the closer $E_{i,d}$ to the interval center, the smaller the perturbation. As a result, Eq. (6) adaptively determines the local search granularity based on the interval size and the distance between the exemplar position and the interval center on the dimension to be updated. Because Eq. (6) intentionally pulls each particle towards some position other than the exemplar position on the dth dimension, a hopeful better solution around the interval is expected to be found. If the inertia weight w_{PbE} is determined in a linear decreasing manner according to (3), w_{PbE} may have large and small values during the run. A large w_{PbE} would lead to slow solution refinement, whereas a small w_{PbE} would not be able to introduce enough diversity into exploitation. On the contrary, for the perturbation coefficient η, it would be better to have both small and large values for η. A small η contributes to search within or near the normative interval, whereas a large η is helpful to discover the global optimum if the optimum is located outside the normative interval to some extent. Empirically, w_{PbE} is fixed at 0.5, and η is randomly generated from a normal distribution with mean 1 and standard deviation 0.65. Note that η is clamped to 10 times of its standard deviation on both sides of its mean for the purpose of preventing the use of too large values. The perturbation based particle velocity/position update thus contributes to sufficient exploitation around the small normative interval $[\underline{P_d}, \overline{P_d}]$. Fig. 1 illustrates the working principle of the perturbation based exploitation (PbE) strategy.

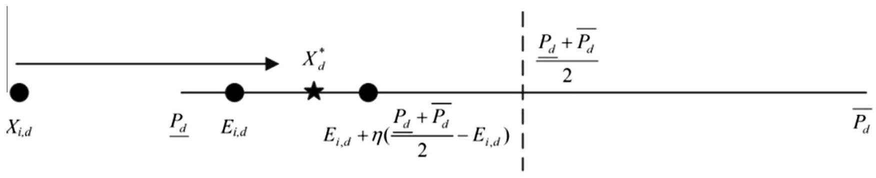
X_d^*
$\frac{\underline{P_d}+\overline{P_d}}{2}$
$X_{i,d}$
$\underline{P_d}$
$E_{i,d}$
$E_{i,d}+\eta(\frac{\underline{P_d}+\overline{P_d}}{2}-E_{i,d})$
$\overline{P_d}$

Fig. 1. Illustration of the perturbation based local search process.

The PbE strategy is essentially different from the perturbed PSO proposed in [34], as the perturbation was applied to the global best position in [34] and the purpose of the perturbed PSO was to maintain diversity during particularly the exploration stage.

Compared with other local search techniques such as Newton's, quasi-Newton, direct search, and chaotic local search [10], the PbE strategy has two unique strengths: first, it is more flexible as it does not require the objective function to be differentiable and there is no need to calculate any derivative; and second, for many real-world problems that are associated with a complex objective function or use some numerical simulation procedure (e.g. finite element) to evaluate the performance of a solution, most of the algorithm computation time is spent on function evaluations (FEs), as the PbE strategy integrates local search into the velocity update procedure, it incurs no additional FE that works outside the PSO framework and is thus significantly more efficient than the other local search techniques.

3.2. Adaptive learning probabilities

In CLPSO, according to (4), a particle's learning probability depends only on its index and the probability value does not change during the entire search process. This static strategy for setting the learning probabilities might hinder convergence [8]. Without loss of generality, suppose the optimization problem on hand is a minimization problem. Our intuition here is that a position with a small fitness value usually is better on more dimensions than a position with a large fitness value and this intuition is true for most cases. Following this intuition, for any two particle i and j with the personal best fitness value of i (i.e. $f(P_i)$) less than that of j (i.e. $f(P_j)$), where $f(\ )$ is a function or numerical simulation procedure that outputs the fitness value of a given position, P_i often is better on more dimensions than P_j. Therefore, when assigning learning probabilities to i and j, L_i needs to be less than L_j as i would benefit more from learning from its own personal best position than j. Accordingly, with the aim of facilitating convergence, we propose an adaptive learning probabilities (ALPs) strategy based on rankings of the personal best fitness values. The strategy works as follows.

First, sort all the particles in ascending order of the personal best fitness values. For each particle i, denote its rank in the sorted sequence as K_i. If a particle has the smallest personal best fitness value, then its rank K_i is 1; if it has the largest personal best fitness value, then its rank is N.

Second, determine the learning probability L_i of each particle i according to (7). In this way, the particles also exhibit different exploration and exploitation capabilities, and this strategy is more in line with our intuition.

$$L_i = L_{\min} + (L_{\max} - L_{\min})\frac{\exp\left(\frac{10(K_i-1)}{N-1}\right) - 1}{\exp(10) - 1} \tag{7}$$

However, the above strategy might suffer from a weakness in case $L_{\max}$ is unsuitably large. If the personal best positions with small fitness values are located in some regions far from the global optimum, then the strategy tends to make all the particles move close to some local optima. For a particle that is large in personal best fitness value but has discovered some dimensions relevant to the global optimum region, chances for selecting the particle for exemplar guidance are small because of the tournament selection procedure, and the particle is likely to learn from the small fitness positions because it has a large learning probability. As a result, the valuable information about the global optimum would be lost soon and the algorithm sometimes might get stuck in premature convergence. Indeed, a small $L_{\max}$ benefits exploration, whereas a large $L_{\max}$ contributes to exploitation. To make a tradeoff of avoiding premature convergence and facilitating convergence, the ALPs strategy is modified through further adaptively updating $L_{\max}$ during the run. Specifically, in each generation k, let M_k be the number of the dimensions whose normative intervals have ever satisfied the exploitation activation condition specified in (5) before or just in the generation k, the maximum learning probability $L_{\max}$ is empirically determined according to (8).

$$L_{\max} = L_{\min} + 0.25 + 0.45\log_{(D+1)}(M_k + 1) \tag{8}$$

Note that $L_{\min}$ is fixed at 0.05. As can be seen, $L_{\max}$ is small (i.e. 0.3) when $M_k = 0$ and thus helps to explore the search space incorporating the concern of avoiding premature convergence. $L_{\max}$ increases logarithmically with the particles' exploitation progress, and the more dimensions undergoing exploitation, the larger $L_{\max}$ to facilitate convergence. $L_{\max}$ may reach the largest value 0.75 if $M_k = D$. The logarithmic function $\log_{(D+1)}(M_k + 1)$ used in (8) has a higher increment rate than the linear function M_k/D.

3.3. Flow chart and computation requirements analysis of ECLPSO

The flow chart of ECLPSO is summarized as follows.

Step 1 Initialize the velocities and positions; evaluate the fitness values and determine the personal best positions; set the generation counter $k = 1$, the stagnation number of each particle $S_i = 0$, and the other algorithm parameters.
Step 2 Is $k \leqslant k_{\max}$? If yes, go to step 3; otherwise, go to step 19.
Step 3 Compute the normative knowledge and calculate the number of exploitation valid dimensions M_k.

Step 4 If there exists at least one particle that its exemplar positions need to be re-assigned (i.e. its stagnation number = 0), determine the learning probabilities of all the particles using the ALPs strategy.
Step 5 Update the inertia weight w.
Step 6 Initialize the particle counter $i = 1$.
Step 7 Is $i \leqslant N$? If yes, go to step 8; otherwise, go to step 18.
Step 8 If the stagnation number $S_i = 0$, re-assign the exemplar positions for i.
Step 9 Initialize the dimension counter $d = 1$.
Step 10 Is $d \leqslant D$? If yes, go to step 11; otherwise, go to step 14.
Step 11 If Eq. (5) is false, update the dimensional velocity $V_{i,d}$ according to (1); otherwise, update the dimensional velocity according to (6).
Step 12 Update the dimensional position $X_{i,d}$.
Step 13 Increase the dimension counter d by 1, and go back to step 10.
Step 14 Is the position X_i within the search space? If yes, go to step 15; otherwise, go to step 17.
Step 15 Evaluate the fitness value of particle i, i.e. $f(X_i)$.
Step 16 If $f(X_i) < f(P_i)$, update the personal best position P_i; otherwise, set the stagnation number $S_i = (S_i + 1) \bmod g$.
Step 17 Increase the particle counter i by 1, and go back to step 7.
Step 18 Increase the generation counter k by 1, and go back to step 2.
Step 19 Output the final global best position.

As can be seen in step 11, with the PbE strategy employed, the particle velocity update procedure takes different forms during the exploration and exploitation stages.

ECLPSO needs to store algorithm related data structures such as the velocities, positions, fitness values, personal best positions and fitness values, fitness rankings, learning probabilities, exemplar assignments, stagnation numbers, normative knowledge, and the historical exploitation status of each dimension. Storing these data structures requires $O(ND)$ memory space.

Computing the normative knowledge in step 3 requires $O(ND)$ comparisons. The sorting of the personal best fitness values in step 4 requires $O(N\log N)$ comparisons and swaps. The other steps in one generation require $O(ND)$ basic operations. Step 15 requires one FE. Step 1 is only invoked once. As there are $k_{\max}$ generations, the time requirement of ECLPSO is $O(k_{\max}(N\log N + ND))$ basic operations plus $O(k_{\max}N)$ FEs.

4. Experimental results

4.1. Test functions

Fourteen test functions widely adopted in benchmarking global optimization algorithms [6,35] are used in our experiment. The functions are classified into three general categories: unimodal, multimodal, and rotated. The expressions of the functions are presented in Table 2. The dimension D is equivalent to 30 in our experiment.

The functions f_1, f_2, f_3, and f_4 are unimodal. The sphere and Schwefel's P2.22 functions are simple unimodal. The Rosenbrock's function is unimodal in a two or three-dimensional search space but can be treated as a multimodal function in high-dimensional cases. The Rosenbrock's function has a narrow valley from perceived local optima to the global optimum. Noisy perturbation is introduced to the noise function.

The functions f_5, f_6, f_7, f_8, f_9, and f_{10} are multimodal. The Schwefel's function has deep local optima that are located far from the global optimum. There are a large number of local optima for the Rastrigin's function. The Ackley's function has a narrow global optimum region and many minor local optima. The Griewank's function has a sinusoidal multiplication term that causes linkages among the variables.

The functions f_{11}, f_{12}, f_{13}, and f_{14} are rotated. To rotate a function, first an orthogonal matrix M is generated [36]. The original variable vector x is left multiplied by M to get a new rotated variable vector $y = Mx$. Owing to the linear summation caused by the rotation, if one dimension of x is changed, all the dimensions in y will be affected.

Table 2 also lists the global optimum x^*, corresponding fitness value $f(x^*)$, search space, and initialization space of each function. Biased initializations are used for the functions whose global optimum is located at the center of the search space.

4.2. Particle swarm optimization algorithms compared and performance metrics

In this paper, the following two performance issues are studied: first, how the proposed PbE and ALPs enhancements improve the algorithm performance; and second, how ECLPSO performs compared with other PSO algorithms. For the first issue, four more ECLPSO variants, namely ECLPSO-1, ECLPSO-2, ECLPSO-3, and ECLPSO-4 are studied. Neither ECLPSO-1 nor ECLPSO-2 adopts the PbE enhancement, whereas both ECLPSO-3 and ECLPSO-4 take advantage of the PbE enhancement. The ALPs enhancement is used in ECLPSO-1, ECLPSO-2, and ECLPSO-4, whereas ECLPSO-3 still relies on the original static strategy for setting the particles' learning probabilities. $L_{\max}$ is fixed at 0.3 in all the generations in ECLPSO-1 and ECLPSO-4, while is allowed to adaptively increase to 0.75 in ECLPSO-2. Regarding the second issue, ECLPSO is compared with CLPSO, OLPSO-G,

Table 2
Expressions, global optima, and search/initialization spaces of the test functions.

Function	Description and expression	x^*	$f(x^*)$	Search space	Initialization space
f_1 [35]	Sphere, $f_1(x) = \sum_{d=1}^{D} x_d^2$	$\{0\}^D$	0	$[-100, 100]^D$	$[-100, 50]^D$
f_2 [35]	Schwefel's P2.22, $f_2(x) = \sum_{d=1}^{D} \lvert x_d\rvert + \prod_{d=1}^{D} \lvert x_d\rvert$	$\{0\}^D$	0	$[-10, 10]^D$	$[-10, 5]^D$
f_3 [35]	Rosenbrock's, $f_3(x) = \sum_{d=1}^{D-1}\left(100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2\right)$	$\{1\}^D$	0	$[-10, 10]^D$	$[-10, 10]^D$
f_4 [35]	Noise, $f_4(x) = \sum_{d=1}^{D} d x_d^4 + \text{random}\,[0, 1)$	$\{0\}^D$	0	$[-1.28, 1.28]^D$	$[-1.28, 0.64]^D$
f_5 [35]	Schwefel's, $f_5(x) = 418.9829D - \sum_{d=1}^{D} x_d \sin\left(\sqrt{\lvert x_d\rvert}\right)$	$\{420.96\}^D$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_6 [35]	Rastrigin's, $f_6(x) = \sum_{d=1}^{D} (x_d^2 - 10\cos(2\pi x_d) + 10)$	$\{0\}^D$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_7 [35]	Ackley's, $f_7(x) = -20\exp\left(-0.2\sqrt{\left(\sum_{d=1}^{D} x_d^2\right)/D}\right) - \exp\left(\left(\sum_{d=1}^{D}\cos(2\pi x_d)\right)/D\right) + 20 + e$	$\{0\}^D$	0	$[-32, 32]^D$	$[-32, 16]^D$
f_8 [35]	Griewank's, $f_8(x) = \sum_{d=1}^{D} (x_d^2/4000) - \prod_{d=1}^{D}\cos(x_d/\sqrt{d}) + 1$	$\{0\}^D$	0	$[-600, 600]^D$	$[-600, 200]^D$
f_9 [35]	A generalized penalized, $f_9(x) = \pi\left(10\sin^2(\pi y_1) + \sum_{d=1}^{D-1}((y_d - 1)^2(1 + 10\sin^2(\pi y_{d+1}))) + (y_D - 1)^2\right)/D + \sum_{d=1}^{D} u(x_d, 10)$, where $y_d = 1 + (x_d + 1)/4$, $u(x_d, a) = \begin{cases} 100(x_d - a)^4, & \text{if } x_d > a \\ 0, & \text{if } -a \leqslant x_d \leqslant a \\ 100(-x_d - a)^4, & \text{if } x_d < -a \end{cases}$	$\{0\}^D$	0	$[-50, 50]^D$	$[-50, 25]^D$
f_{10} [35]	Another generalized penalized, $f_{10}(x) = \left(\sin^2(3\pi x_1) + \sum_{d=1}^{D-1}((x_d - 1)^2(1 + \sin^2(3\pi x_{d+1}))) + (x_D - 1)^2(1 + \sin^2(2\pi x_D))\right)/10 + \sum_{d=1}^{D} u(x_d, 5)$	$\{0\}^D$	0	$[-50, 50]^D$	$[-50, 25]^D$
f_{11} [6]	Rotated Schwefel's, $f_{11}(y) = 418.9828D - \sum_{d=1}^{D} z_d$, where $z_d = \begin{cases} y_d \sin\left(\sqrt{\lvert y_d\rvert}\right), & \text{if } \lvert y_d\rvert \leqslant 500 \\ 0, & \text{otherwise} \end{cases}$, $y_d = y'_d + 420.96$, where $y' = M(x - 420.96)$, M is an orthogonal matrix	$\{420.96\}^D$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{12} [6]	Rotated Rastrigin's, $f_{12}(y) = f_6(y)$, where $y = Mx$	$\{0\}^D$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{13} [6]	Rotated Ackley's, $f_{13}(y) = f_7(y)$, where $y = Mx$	$\{0\}^D$	0	$[-32, 32]^D$	$[-32, 16]^D$
f_{14} [6]	Rotated Griewank's, $f_{14}(y) = f_8(y)$, where $y = Mx$	$\{0\}^D$	0	$[-600, 600]^D$	$[-600, 200]^D$

OLPSO-L, and SPSO. All the algorithms are tested using the same swarm size of 40. All the algorithms use the same maximum number of FEs 200,000 in each run on each function. Each algorithm is tested 25 times independently on each function. For CLPSO and the ECLPSO variants, the algorithm parameters take the recommended empirical values stated in previous sections. The results data of OLPSO-G, OLPSO-L, and SPSO are obtained from [7].

The following performance metrics are used to evaluate the algorithms' performance on each function: (1) the mean, standard deviation (SD), worst, median, and best of the final solutions obtained from the 25 runs; and (2) the average number of exploitation valid dimensions (EVDs) of the 25 runs.

4.3. Results and discussions

Table 3 lists mean and SD results of the 25 runs of all the algorithms on the unimodal and multimodal functions. Table 4 lists mean and SD results of CLPSO and the ECLPSO variants on the rotated functions. The best results among the algorithms are marked in bold in Tables 3 and 4. The orthogonal matrix M that is left multiplied to determine the rotated variables in a rotated function is generated using the Salomon's method [36]. For all the rotated functions, the alpha value used in the "MRot" procedure of the Salomon's method is fixed at 0.1. Because we do not know what exact orthogonal matrices were used in the experiment conducted in [7], the algorithms OLPSO-G, OLPSO-L, and SPSO are not included in the performance comparison on the rotated functions. Tables 5 and 6 list the other metrics results of CLPSO and the ECLPSO variants on all the functions. Table 7 reports results of ECLPSO-1 and ECLPSO-4 using some different parameter settings. To determine whether the solutions obtained by CLPSO are statistically different from those by ECLPSO in terms of mean and SD, two-tailed t-tests with degrees of freedom 48 and significance level 0.05 are carried out and the t-test results are listed in Table 8. Tables 9 and 10 list mean and SD results of ECLPSO on f_1 using different values for η and w_{PbE}. Fig. 2 illustrates convergence characteristics of CLPSO and ECLPSO in terms of the global best fitness values of a median run on f_1, f_2, f_6, f_7, f_8, f_9, f_{10}, f_{13}, and f_{14}.

(1) Comparison of the static and ALPs strategies: Observed from Tables 3 and 4, ELPSO-1 performs better than CLPSO on f_3, f_8, f_{11}, f_{12}, and f_{14}, and ties with CLPSO on the other functions, in terms of the mean and SD results. It is thus clear that the ALPs strategy is a better alternative to the static strategy as the ALPs strategy assigns appropriate learning probabilities to the particles based on the rankings of the personal best fitness values. If we fix L_{max} at 0.75, ECLPSO-1 sometimes is liable to get stuck in a local optimum on f_5, as indicated from the worst and mean results in Table 7. Comparing the results of ECLPSO-1 on f_5 with $L_{\max}$ fixed at 0.3 and 0.75, respectively, we can see that a small $L_{\max}$ helps

Table 3
Mean and SD results of all the algorithms on the unimodal and multimodal test functions. The best results among the algorithms are marked in bold.

Test function		CLPSO	ECLPSO-1	ECLPSO-2	ECLPSO-3	ECLPSO-4	ECLPSO	OLPSO-G	OLPSO-L	SPSO
f_1	Mean	2.56E−14	2.49E−14	2.68E−15	5.29E−82	8.41E−92	**1.00E−96**	4.12E−54	1.11E−38	2.29E−96
	SD	8.77E−14	5.81E−14	8.49E−15	2.52E−81	2.14E−91	**3.01E−96**	6.34E−54	1.28E−38	9.48E−96
f_2	Mean	3.12E−10	2.80E−10	2.48E−11	3.06E−28	9.01E−30	2.02E−31	9.85E−30	7.67E−22	**1.74E−53**
	SD	1.96E−10	1.69E−10	7.84E−11	4.29E−28	8.63E−30	2.84E−31	1.01E−29	5.63E−22	**1.58E−53**
f_3	Mean	39.17	27.46	27.46	39.17	27.46	27.46	21.52	**1.26**	13.50
	SD	21.31	15.03	15.03	21.31	15.03	15.03	29.92	**1.40**	14.63
f_4	Mean	4.91E−3	5.66E−3	5.66E−3	4.91E−3	5.66E−3	5.66E−3	1.16E−2	1.64E−2	**4.02E−3**
	SD	1.11E−3	1.03E−3	1.03E−3	1.11E−3	1.03E−3	**1.03E−3**	4.10E−3	3.25E−3	1.66E−3
f_5	Mean	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	**3.82E−4**	3.84E2	3.82E−4	3.14E3
	SD	2.42E−13	3.92E−12	5.64E−12	0	0	**0**	2.17E2	0	7.81E2
f_6	Mean	1.94E−6	2.82E−6	1.74E−6	1.92E−11	0	**0**	1.07	0	41.03
	SD	1.74E−6	2.97E−6	2.16E−6	8.81E−11	0	**0**	0.99	0	11.09
f_7	Mean	3.20E−8	6.33E−8	6.34E−9	3.69E−15	3.55E−15	**3.55E−15**	7.98E−15	4.14E−15	3.73E−2
	SD	5.28E−8	7.32E−8	7.84E−9	7.11E−16	0	**0**	2.03E−15	0	0.19
f_8	Mean	1.32E−9	1.56E−10	2.16E−12	7.95E−11	5.36E−10	**0**	4.83E−3	0	7.48E−3
	SD	2.70E−9	2.89E−10	1.00E−11	2.20E−10	1.31E−9	**0**	8.63E−3	0	1.25E−2
f_9	Mean	1.94E−16	3.15E−16	1.59E−17	1.57E−32	1.57E−32	**1.57E−32**	1.59E−32	1.57E−32	7.47E−2
	SD	1.97E−16	3.76E−16	7.13E−17	8.38E−48	8.38E−48	8.38E−48	1.03E−33	**2.79E−48**	3.11
f_{10}	Mean	1.11E−13	3.44E−13	2.34E−14	1.37E−32	1.51E−32	**1.35E−32**	4.39E−4	1.57E−32	1.76E−3
	SD	1.60E−13	5.25E−13	5.37E−14	9.86E−34	4.20E−33	2.47E−34	2.20E−3	**2.79E−48**	4.11E−3

Table 4
Mean and SD results of CLPSO and the ECLPSO variants on the rotated test functions.

Test function		CLPSO	ECLPSO-1	ECLPSO-2	ECLPSO-3	ECLPSO-4	ECLPSO
f_{11}	Mean	1.28E3	1.16E3	1.16E3	1.28E3	1.16E3	**1.16E3**
	SD	1.20E2	1.44E2	1.44E2	1.20E2	1.44E2	**1.44E2**
f_{12}	Mean	30.90	22.70	22.70	30.90	22.70	**22.70**
	SD	4.49	4.47	4.47	4.49	4.47	**4.47**
f_{13}	Mean	5.63E−8	5.30E−8	7.42E−9	3.69E−15	3.55E−15	**3.55E−15**
	SD	3.46E−8	3.51E−8	7.08E−9	7.11E−16	0	**0**
f_{14}	Mean	3.58E−5	5.94E−6	1.07E−7	2.07E−3	9.75E−4	**2.22E−17**
	SD	4.71E−5	7.22E−6	2.86E−7	2.09E−3	8.94E−4	**4.53E−17**

Table 5
Worst, median, and best results of CLPSO, ECLPSO-3, ECLPSO-4, and ECLPSO on all the test functions.

Test function	CLPSO			ECLPSO-3			ECLPSO-4			ECLPSO		
	Worst	Median	Best	Worst	Median	Best	Worst	Median	Best	Worst	Median	Best
f_1	4.45E−13	4.49E−15	5.88E−16	1.26E−80	2.40E−85	4.30E−90	1.05E−90	6.93E−93	7.28E−95	1.49E−95	1.25E−97	2.11E−100
f_2	8.66E−10	2.61E−10	1.32E−10	1.73E−27	1.56E−28	2.74E−29	3.26E−29	7.31E−30	5.71E−31	1.11E−30	8.84E−32	1.49E−32
f_3	81.22	28.03	17.49	81.22	28.03	17.49	76.75	24.71	17.17	76.75	24.71	17.17
f_4	7.37E−3	4.98E−3	3.98E−3	7.37E−3	4.98E−3	3.98E−3	7.64E−3	5.65E−3	3.62E−3	7.64E−3	5.65E−3	3.62E−3
f_5	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4	3.82E−4
f_6	6.90E−6	1.26E−6	2.00E−7	4.40E−10	0	0	0	0	0	0	0	0
f_7	2.78E−7	2.03E−8	5.71E−9	7.11E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15
f_8	1.06E−8	1.67E−10	5.90E−12	1.08E−9	6.63E−12	5.34E−14	6.13E−9	2.24E−13	0	0	0	0
f_9	7.55E−16	1.47E−16	6.62E−18	1.57E−32	1.57E−32	1.57E−32	1.57E−32	1.57E−32	1.57E−32	1.57E−32	1.57E−32	1.57E−32
f_{10}	7.22E−13	4.48E−14	4.53E−15	1.84E−32	1.35E−32	1.35E−32	3.32E−32	1.35E−32	1.35E−32	1.47E−32	1.35E−32	1.35E−32
f_{11}	1.46E3	1.32E3	9.48E2	1.46E3	1.32E3	9.48E2	1.49E3	1.17E3	8.53E2	1.49E3	1.17E3	8.53E2
f_{12}	41.21	29.89	24.06	41.21	29.89	24.06	29.59	23.41	15.73	29.59	23.41	15.73
f_{13}	1.38E−7	5.34E−8	1.96E−8	7.11E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15	3.55E−15
f_{14}	1.63E−4	1.20E−5	9.19E−7	1.00E−2	1.31E−3	2.22E−4	3.49E−3	7.56E−4	0	1.11E−16	0	0

to explore the search space and avoid premature convergence. As the EVDs results in Table 6 show, in average the particles reach an equilibrium state on equivalent or more number of dimensions in ECLPSO-1 than in CLPSO on all the functions, demonstrating that the ALPs strategy is able to facilitate convergence.

(2) Combined use of the PbE and ALPs enhancements: As can be seen from Tables 3, 4 and 6, the mean and SD results of ECLPSO-2 are slightly better than those of ECLPSO-1 on $f_1, f_2, f_7, f_8, f_9, f_{10}$, and f_{14}, and same on f_3, f_4, f_{11}, and f_{12} (because the EVDs results are 0 on the four functions). The EVDs results of ECLPSO-2 are no less than those of ECLPSO-1, and remarkably more on f_8 and f_{14}. The observations verify that a large $L_{\max}$ benefits exploitation. The mean and SD results

Table 6
EVDs results of CLPSO and the ECLPSO variants on all the test functions.

		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
CLPSO	EVDs	28	29	0	0	20	17	29	17	29	30	0	0	29	13
ECLPSO-1		30	30	0	0	20	22	29	20	30	30	0	0	29	19
ECLPSO-2		30	30	0	0	21	22	29	26	30	30	0	0	30	27
ECLPSO-3		30	30	0	0	30	30	30	19	30	30	0	0	30	15
ECLPSO-4		30	30	0	0	30	30	30	28	30	30	0	0	30	22
ECLPSO		30	30	0	0	30	30	30	30	30	30	0	0	30	30

Table 7
Results of ECLPSO-1 and ECLPSO-4 using some different parameter settings.

	Test function	Mean	SD	Worst	Median	Best	EVDs
ECLPSO-1 with $L_{\max}$ fixed at 0.75	f_5	14.21	52.08	236.88	3.82E−4	3.82E−4	30
ECLPSO-4 with $L_{\max}$ fixed at 0.75 when $M_k > 0$[a]	f_6	3.98E−2	1.99E−1	9.95E−1	0	0	30

[a] $L_{\max}$ is fixed at 0.3 when $M_k = 0$.

Table 8
Two-tailed t-test values from the comparison of CLPSO and ECLPSO on all the test functions.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
t-test	1.46	7.95[b]	2.25[b]	−2.51[b]	4.32[b]	5.58[b]	3.03[b]	2.45[b]	4.93[b]	3.47[b]	3.19[b]	6.47[b]	8.15[b]	3.80[b]

[b] The value is significant by a two-tailed t-test between CLPSO and ECLPSO.

Table 9
Mean and SD results of ECLPSO using different standard deviations for η on f_1.

Standard deviation of η	0.8	0.7	0.65	0.6
Mean	1.05E−52	1.32E−80	1.00E−96	7.76E−5
SD	1.34E−52	2.14E−80	3.01E−96	2.72E−4

Table 10
Mean and SD results of ECLPSO using different inertia weights in the PbE strategy on f_1.

w_{PbE}	Linear decreasing	Fixed at 0.6	Fixed at 0.55	Fixed at 0.5	Fixed at 0.45	Fixed at 0.4
Mean	5.89E−46	7.14E−52	3.75E−74	1.00E−96	3.67E−6	6.86E−3
SD	2.95E−45	2.25E−51	1.43E−73	3.01E−96	1.81E−5	1.51E−2

of ECLPSO-2, ECLPSO-3, and ECLPSO are much better than those of ECLPSO-2 on $f_1, f_2, f_6, f_7, f_9, f_{10}$, and f_{13}. Only the EVDs results of ECLPSO are 30 on f_8 and f_{14}. In terms of the mean and SD results, ECLPSO beats ECLPSO-3 and ECLPSO-4 on all the functions, ECLPSO-4 is overall better than ECLPSO-3 with f_4, f_8, and f_{10} being exceptions. The worst, median, and best results in Table 5 indicate that CLPSO cannot reach the global optimum on every function and in every run, whereas ECLPSO finds the global optimum on f_6, f_8, and f_{14} in most or all of the 25 runs. It can be derived from the t-test values in Table 6 that the solutions obtained by ECLPSO are statistically better than those obtained by CLPSO on 12 functions. The convergence processes illustrated in Fig. 2 shows remarkable improvements of the global best fitness value during the exploitation stage in all the median runs of ECLPSO on f_1, f_2, f_6, f_7, f_8, f_9, f_{10}, f_{13}, and f_{14}. As observed from Table 7, if we fix L_{max} at 0.75 when $M_k > 0$, ECLPSO-4 sometimes loses its way at a local optimum on f_6. The many observations clarify the following aspects: (i) the PbE enhancement can realize high performance exploitation and thus improve solution accuracy; (ii) the combination of the PbE and ALPs enhancements is able to achieve further improvement on solution accuracy; (iii) a large $L_{\max}$ is needed in the exploitation stage to facilitate convergence; and (iv) gradually increasing $L_{\max}$ with the growth of M_k is preferable over the alternative of simply setting $L_{\max}$ at a fixed large value when $M_k > 0$ because the latter approach is likely to cause premature convergence.

(3) Rotated problems: For rotated problems, a small change of the original variable vector on one dimension causes changes of the rotated variable vector on all the dimensions. Hence, there could be a lot of points with large fitness values around the small global optimum region. Because of the use of the PbE strategy, ECLPSO-3 and ECLPSO-4 search more diversely during the exploitation stage, and they might find a worse solution than CLPSO on f_{14}, as can be seen from the worst results on f_{14} given in Table 5. However, as ECLPSO discourages learning from the personal best posi-

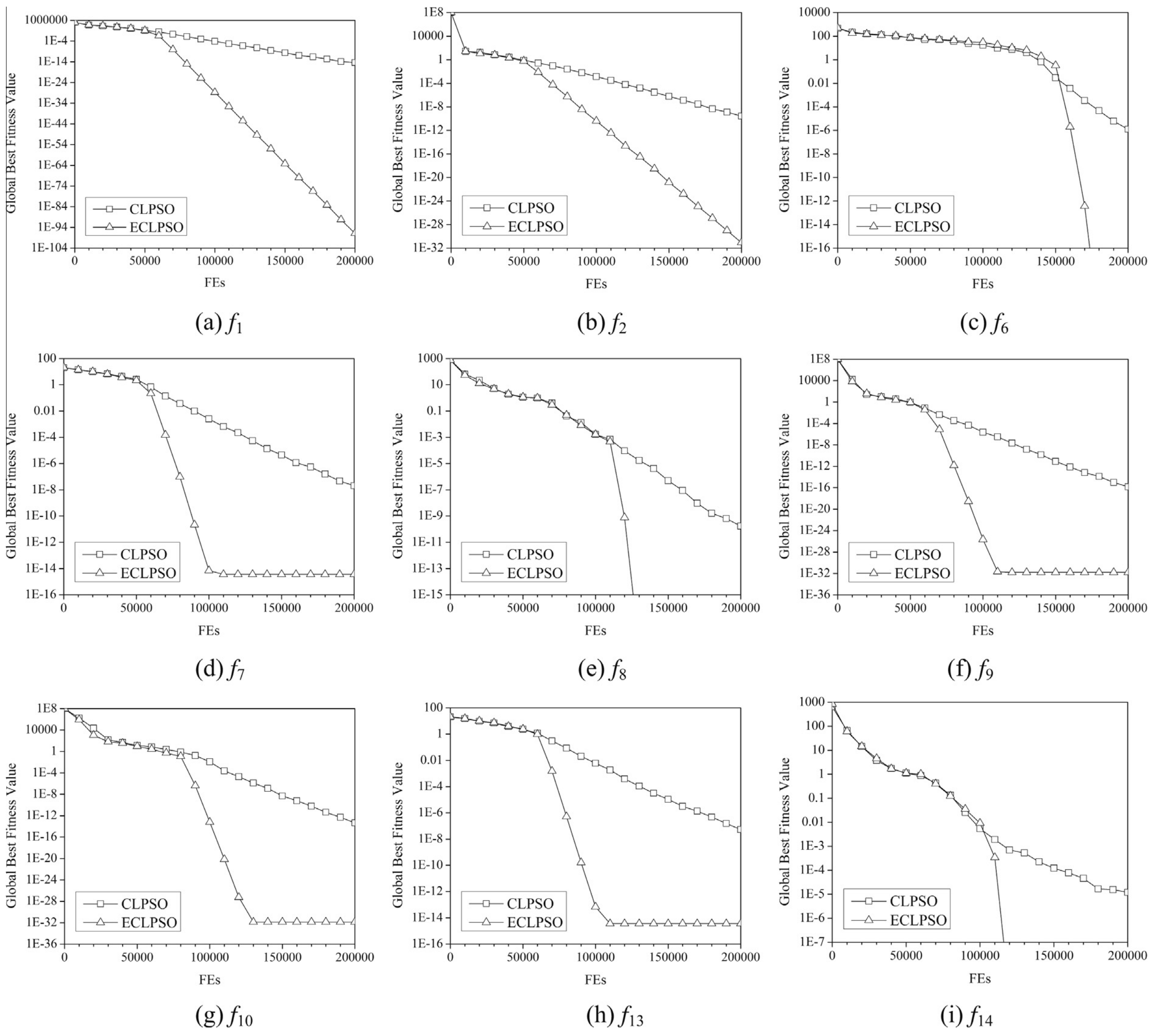

Fig. 2. Convergence characteristics of CLPSO and ECLPSO on f_1, f_2, f_6, f_7, f_8, f_9, f_{10}, f_{13}, and f_{14}.

tions with large fitness values, the solutions obtained by ECLPSO are significantly better than those obtained by CLPSO on f_{14}. CLPSO and the ECLPSO variants cannot activate a local search on any dimension on f_{11} and f_{12} because there are many local optimum regions scattered in the search space.

(4) Comparison of the PSO algorithms on unimodal and multimodal problems: In terms of the mean results given in Tables 3 and 4, SPSO beats all the other algorithms on f_2 and f_4, OLPSO-L performs the best on f_3, and ECLPSO is the winner on all the other functions. OLPSO-G performs better than OLPSO-L on f_1 and f_2. Because SPSO and OLPSO-G are not good at preserving the swarm's diversity, they do not perform well on complex multimodal functions. Owing to the random noise introduced into f_4, OLPSO-G and OLPSO-L perform badly on f_4, and no dimensional local search has been activated in ECLPSO on f_4. The mean results of ECLPSO are better than OLPSO-L on f_1, f_2, f_4, f_7, and f_{10}, equivalent on f_5, f_6, f_8, and f_9, and worse on f_3. The function f_3 features a narrow valley from perceived local optima to the global optimum and thus ECLPSO tends to be troubled by premature convergence on f_3. To solve f_3, ECLPSO can incorporate techniques such as reinitialization [22], subgrouping [24], perturbation [34], mutation [17,18], and chaotic dynamics [10,25] to continuously inject randomness into the swarm so as to increase the swarm's diversity, until the algorithm has located the global optimum region. The EVDs of ECLPSO on f_3 is 0, meaning that because of the narrow valley, the personal best positions span across the valley on all the dimensions.

(5) Tuning of the algorithm parameters: Several new parameters are introduced into ECLPSO. The appropriate values of the parameters are determined based on trials on all the functions. Tables 9 and 10 are examples to indicate that the algorithm performance is sensitive to the values of the algorithm parameters.

(6) Discussions: The experimental results and comparison demonstrate that ECLPSO benefits from the two enhancements, i.e. PbE and ALPs. Indeed, ECLPSO performs significantly better than CLPSO in terms of solution accuracy. The PbE and ALPs enhancements are able to utilize valuable information embedded in the particles' search experiences. ECLPSO performs better than OLPSO-L on the unimodal and multimodal functions. According to [7], the implementation of OLPSO-L involves constructing a static ring topology and determining the guidance vector through a rather complicated orthogonal experimental design. Like CLPSO, OLPSO-L also needs to re-determine exemplars after a consecutive number of generations. ECLPSO is easier to implement than OLPSO-L as ECLPSO just introduces a few simple modifications to CLPSO. For the orthogonal experimental design, OLPSO-L needs to store an orthogonal array using $O\left(2^{\lceil \log_2(D+1)\rceil}D\right)$ memory space. The factor analysis conducted by each particle in OLPSO-L requires $O\left(2^{\lceil \log_2(D+1)\rceil}\right)$ FEs and $O\left(2^{\lceil \log_2(D+1)\rceil}D\right)$ basic operations. Therefore, OLPSO-L needs $O\left(2^{\lceil \log_2(D+1)\rceil}D + ND\right)$ memory space to store its algorithm related data structures and the time requirement of OLPSO-L is $O\left(k_{\max}N2^{\lceil \log_2(D+1)\rceil}D\right)$ basic operations plus $O\left(k_{\max}N2^{\lceil \log_2(D+1)\rceil}\right)$ FEs. For many real-world problems, D is far larger than N. For example, unit commitment is a critical issue related to daily/weekly planning of modern power systems. Zhao et al. [37] solved a daily unit commitment problem of a 100-unit power system with 2400 decision variables using PSO. The commitment of 40 units in the Taipower system over a weekly planning horizon studied in [38] involves 6720 decision variables. Hence, for such real-world problems, OLPSO-L requires considerably more memory space than ECLPSO, and the two algorithms' time requirements on the basic operations other than FEs will be in the same scale, if the algorithms run for the same number of FEs. Based on the discussions, ECLPSO is recommended as a promising global optimizer.

5. Conclusions

In this paper, we have proposed an enhanced method called ECLPSO to supersede the state-of-the-art CLPSO. ECLPSO is integrated with two enhancements to improve the algorithm performance in terms of solution accuracy. Experiments have been conducted on fourteen benchmark functions including unimodal, multimodal, and rotated. The results demonstrate that ECLPSO significantly outperforms CLPSO on the functions tested and is even superior to OLPSO-L. ECLPSO is easier to implement than OLPSO-L. Moreover, ECLPSO requires much less memory space than OLPSO-L for many real-world large-scale problems.

Acknowledgment

This work has been financially supported by the Public Benefit Special Research Fund of the Ministry of Water Resources of China (201201017).

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948.
[2] Y.H. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Piscataway, NJ, 1998, pp. 69–73.
[3] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Honolulu, Hawaii, 2002, pp. 1671–1676.
[4] Particle swarm central, <http://www.particleswarm.info>.
[5] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (2004) 204–210.
[6] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[7] Z.H. Zhan, J. Zhang, Y. Li, Y.H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (2011) 832–847.
[8] J.J. Liang, P.N. Suganthan, Adaptive comprehensive learning particle swarm optimizer with history learning, in: Simulated Evolution and Learning, Springer, 2006, pp. 213–220.
[9] Y.J. Zheng, H.F. Ling, Q. Guan, Adaptive parameters for a modified comprehensive learning particle swarm optimizer, Math. Prob. Eng. 2012 (2012).
[10] J.C. Ni, L. Li, F.I. Qiao, Q.D. Wu, A novel memetic algorithm based on the comprehensive learning PSO, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Brisbane, Australia, 2012, pp. 1–8.
[11] V.L. Huang, P.N. Suganthan, J.J. Liang, Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems, Int. J. Intell. Syst. 21 (2006) 209–226.
[12] T.A.A. Victoire, P.N. Suganthan, Improved MOCLPSO algorithm for environmental/economic dispatch, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Singapore, 2007, pp. 3072–3076.
[13] H. Ali, F.A. Khan, Attributed multi-objective comprehensive learning particle swarm optimization for optimal security of networks, Appl. Soft Comput. 13 (2013) 3903–3921.
[14] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, S.H. Chung, Y. Li, Y.H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2013) 241–258.
[15] Z. Beheshti, S.M.H. Shamsuddin, S. Hasan, MPSO: median-oriented particle swarm optimization, Appl. Math. Comput. 219 (2013) 5817–5836.
[16] G. He, N.J. Huang, A new particle swarm optimization algorithm with an application, Appl. Math. Comput. 232 (2014) 521–528.
[17] Y.V. Pehlivanoglu, A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks, IEEE Trans. Evol. Comput. 17 (2013) 436–452.
[18] H. Wang, W.J. Wang, Z.J. Wu, Particle swarm optimization with adaptive mutation for multimodal optimization, Appl. Math. Comput. 221 (2013) 296–305.

[19] Y.C. Lu, J.C. Jan, S.L. Hung, G.H. Hung, Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures, Eng. Optim. 45 (2013) 1251–1271.
[20] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Anchorage, AK, 1998, pp. 84–89.
[21] K.E. Parsopoulos, M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004) 211–224.
[22] X.F. Xie, W.J. Zhang, Z.L. Yang, Dissipative particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Honolulu, HI, 2002, pp. 1456–1461.
[23] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004) 225–239.
[24] B. Al-kazemi, C.I.K. Mohan, Multi-phase generalization of the particle swarm optimization algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, HI, 2002, pp. 489–494.
[25] Y. Feng, G.F. Teng, A.X. Wang, Y.M. Yao, Chaotic inertia weight in particle swarm optimization, in: International Conference on Innovative Computing, Information and Control, Japan, 2007, pp. 475–475.
[26] Z.H. Zhan, J. Zhang, Y. Li, H.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 39 (2009) 1362–1381.
[27] M.Q. Hu, T. Wu, J.D. Weir, An adaptive particle swarm optimization with multiple adaptive methods, IEEE Transactions on Evolutionary Computation, vol. 17, pp. 705–720.
[28] X.C. Zhao, W.Q. Lin, Q.F. Zhang, Enhanced particle swarm optimization based on principal component analysis and line search, Appl. Math. Comput. 229 (2014) 440–456.
[29] X. Jin, Y.Q. Liang, D.P. Tian, F.Z. Zhuang, Particle swarm optimization using dimension selection methods, Appl. Math. Comput. 219 (2013) 5185–5197.
[30] J. Sun, W. Fang, V. Palade, X.J. Wu, W.B. Xu, Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point, Appl. Math. Comput. 218 (2011) 3763–3775.
[31] M. Gang, Z. Wei, C. Xiaolin, A novel particle swarm optimization algorithm based on particle migration, Appl. Math. Comput. 218 (2012) 6620–6626.
[32] K. Deb, N. Padhye, Enhancing performance of particle swarm optimization through an algorithmic link with genetic algorithms, Comput. Optim. Appl. 57 (2014) 761–794.
[33] H. Qin, J.Z. Zhou, Y.L. Lu, Y.H. Li, Y.C. Zhang, Multi-objective cultured differential evolution for generating optimal trade-offs in reservoir flood control operation, Water Resour. Manage 24 (2010) 2611–2632.
[34] X.C. Zhao, A perturbed particle swarm algorithm for numerical optimization, Appl. Soft Comput. 10 (2010) 119–124.
[35] X. Yao, Y. Liu, G.M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[36] R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms, Biosystems 39 (1996) 263–278.
[37] B. Zhao, C.X. Guo, B.R. Bai, Y.J. Cao, An improved particle swarm optimization algorithm for unit commitment, Int. J. Electr. Power Energy Syst. 28 (2006) 482–490.
[38] P.H. Chen, Two-level hierarchical approach to unit commitment using expert system and elite PSO, IEEE Trans. Power Syst. 27 (2012) 780–789.