Information Sciences 181 (2011) 4515–4538

Contents lists available at ScienceDirect

Information Sciences

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

Self-adaptive learning based particle swarm optimization

Yu Wang, Bin Li *, Thomas Weise, Jianyu Wang, Bo Yuan, Qiongjie Tian

4th Mailbox, Heifei, Anhui, University of Science and Technology of China, China

ARTICLE INFO

Article history:
Available online 24 July 2010

Keywords:
Particle swarm
Self-adaptive learning
Numerical optimization
Economic load dispatch
Power system

ABSTRACT

Particle swarm optimization (PSO) is a population-based stochastic search technique for solving optimization problems over continuous space, which has been proven to be efficient and effective in wide applications in scientific and engineering domains. However, the universality of current PSO variants, i.e., their ability to achieve good performance on a variety of different fitness landscapes, is still unsatisfying. For many practical problems, where the fitness landscapes are usually unknown, employing a trial-and-error scheme to search for the most suitable PSO variant is computationally expensive. Therefore, it is necessary to develop a more adaptive and robust PSO version to provide users a black-box tool for various application problems. In this paper, we propose a self-adaptive learning based PSO (SLPSO) to make up the above demerits. SLPSO simultaneously adopts four PSO based search strategies. A probability model is used to describe the probability of a strategy being used to update a particle. The model is self-adaptively improved according to the strategies' ability of generating better quality solutions in the past generations. In order to evaluate the performance of SLPSO, we compare it with eight state-of-the-art PSO variants on 26 numerical optimization problems with different characteristics such as uni-modality, multi-modality, rotation, ill-condition, mis-scale and noise. The experimental results clearly verify the advantages of SLPSO. Moreover, a practical engineering problem, the economic load dispatch problem of power systems (ELD), is used to further evaluate SLPSO. Compared with the previous effective ELD evolutionary algorithms, SLPSO can update the best solution records.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Inspired by the concerted actions of flocks of birds, shoals of fish, and swarms of insects searching for food, Kennedy and Eberhart originally proposed particle swarm optimization (PSO) in the mid-1990s [23,24]. As an important branch of swarm intelligence, PSO has attracted public attention from the research community and has been successfully implemented in various scientific and engineering applications, such as mixed discrete nonlinear programming [38], nearest neighborhood classification [8], receive-diversity-aided STBC systems [30], software development [9], Value-at-Risk based fuzzy random facility location models [60] and economic load dispatch in power systems [45]. These applications can be uniformly formulated as D-dimensional optimization problems over continuous space:

$$\begin{aligned} &\text{minimize } f(x), \\ &\text{where } x = [x_1, x_2, \ldots, x_D]. \end{aligned} \tag{1}$$

* Corresponding author. Tel.: +86 551 3601802; fax: +86 551 3601806.
E-mail addresses: wyustc@mail.ustc.edu.cn (Y. Wang), binli@ustc.edu.cn (B. Li), tweise@ustc.edu.cn (T. Weise), jywang@mail.sitp.ac.cn (J. Wang), yuanbo@mail.ustc.edu.cn (B. Yuan), tianqjie@mail.ustc.edu.cn (Q. Tian).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.07.013

Generally speaking, the features of PSOs in dealing with global optimization problems include the improved capability of solving complex problems, high convergence speed and good generality for different problems. These advantages promoted the use of PSO in many engineering applications [16,55]. In the recent years, several state-of-the-art PSO variants have been proposed, such as fully informed PSO (FIPS-PSO) [36], fitness-distance-ratio based PSO (FDR-PSO) [41], cooperative based PSO (CPSO) [4], efficient population utilization strategy based PSO (EPUS-PSO) [19], and comprehensive learning PSO (CLPSO) [28] to name a few.

Although many variants of PSO have been proposed, the universality and robustness, i.e., the effectiveness of one algorithm in dealing with diverse problems with different characteristics, is still unsatisfying. Generally speaking, the success of PSO in solving one specific problem crucially depends on the choice of suitable strategies, foremost the selection of the velocity updating method. For example, CLPSO [28] is specifically designed for complex multi-modal problems, but the convergence speed of CLPSO on uni-modal problems and relatively simple multi-modal problems is much lower than of the other variants. The PSO variants with high convergence speed, on the other hand, are always prone to shrink towards some local optima in a few generations [19,28]. Therefore, it can be concluded that the conflict between exploration and exploitation is very intense in present PSO variants. Besides, it has been observed that the performance of previous PSO variants on ill-conditioned problems is very poor [19,28,36,41], even when the problems are uni-modal, such as the rotated versions of the classical Schwefel 2.21 problem $f_4(x) = \max |z_i|$ and Rochenbrock problem. The reason for this behavior may be that the velocities of particles in these PSO variants are difficult to be adapted to the narrow valleys towards the global optimum of these problems without losing diversity. However, practical scientific and engineering problems may have any kind of fitness landscape, and furthermore, their structure is usually not known or cannot be analyzed efficiently beforehand. Naturally, employing a trial-and-error scheme to search for a most suitable PSO variant for a specific application problem is one possible approach to handle this problem, but the computational cost of this scheme is usually too high [43]. Therefore, from the perspective of engineering practitioners, it is necessary and urgent to develop a more robust and more effective PSO version that can cope with various problems without resetting the strategies and parameters respectively.

In the previous research of optimization techniques, the scheme of simultaneously using multiple offspring creation strategies from one or more different algorithms has been studied by a number of works [33,34,43,44,46,54,58,59]. In these works, it has been proven that applying multiple strategies or methods in one algorithmic framework is useful for algorithm to obtain good performance on different kinds of problems. In this paper, two new PSO velocity updating strategies, the difference-based velocity updating strategy (DbV) and the estimation-based velocity updating strategy (EbV), are proposed to make up the demerits of PSO on ill-conditioned problems, and to accelerate the convergence on many uni-modal problems, respectively. Then, a self-adaptive learning framework is used to probabilistically steer four PSO velocity updating strategies with different features in parallel to optimize problems with different fitness landscapes. The purpose of using self-adaptive learning framework is to adaptively give preference to appropriate strategies on different problems and at different stages of the optimization process based on the feedback of the quality of solutions generated by them. The fundamental idea of tuning the strategies' contribution proportionally to their previous performance is similar to that of [43,59]. Instead of directly assigning absolute shares of the population to the different strategies according to their effectiveness [44,43,59], SLPSO uses execution probabilities to stochastically determine which strategy is adopted to update the current particle. Besides, instead of using relatively fixed execution probabilities during the whole optimization procedure [54], SLPSO uses a gradual updating mechanism based on a given learning rate to adaptively update the execution probabilities based on the feedback of previous optimization procedure. In order to comprehensively evaluate the performance of SLPSO, we conduct function optimization experiments on 26 numerical optimization problems with different characteristics, such as uni-modality, multi-modality, rotation, ill-condition, mis-scale, and noise. The performance of SLPSO is compared with eight state-of-the-art PSO variants. The experimental results clearly validate the efficiency and effectiveness of SLPSO on diverse problems. Besides, in order to benchmark the robustness of SLPSO on practical engineering problems, we applied it to solve the economic load dispatch problem of power system (ELD). Its performance is compared with that of the optimization algorithms known to be most effective in the ELD domain. The best solution records for several ELD benchmark instances are updated by SLPSO.

The remainder of this paper is structured as follows: In the next section, the original PSO is introduced. Then, a literature review on PSO algorithms is presented by categorizing their features according to three aspects. For each classification aspect, the differences between SLPSO and the previous works are discussed. In Section 3, the DbV and EbV strategies are proposed. In Section 4, the new algorithm SLPSO is proposed and then, the search behaviors of PSO, SLPSO, and CLPSO are analyzed by investigating their search range. Experimental results on function optimization are shown and discussed in the fifth section and in Section 6, the experimental results on large scale ELD problems are presented. Finally, the contributions of this paper are summarized and the future work is outlined in Section 7.

2. Previous work related to PSO

PSO is a population-based, stochastic optimization algorithm based on the idea of a swarm moving over a given landscape. The algorithm adaptively updates the velocities and positions of the members of the swarm by learning from the good experiences. Unlike many other evolutionary algorithms [2,63,67], no mutation or crossover operators are used. In original PSO, the velocity V_i^d and position X_i^d of the dth dimension of the ith particle are updated as follows [23]:

$$V_i^d \leftarrow V_i^d + c_1 \times rand_i^d \times \left(pbest_i^d - X_i^d\right) + c_2 \times rand_i^d \times \left(gbest^d - X_i^d\right), \tag{2}$$

$$X_i^d \leftarrow x_i^d + V_i^d \tag{3}$$

where X_i is the position of the ith particle; V_i stands for the velocity of particle i; $pbest_i$ represents the best location in the search space ever visited by particle i and $gbest$ is the best location discovered so far. The framework of original PSO is shown in Table 1.

In the past few years, many researchers tried different ways to improve the performance of PSO on diverse aspects. In the following, we will discuss three important problems in the PSO domain. The differences between SLPSO and the other works will be emphasized at the end of each sub-section.

2.1. How to adapt suitable parameters?

For the velocity updating strategy of the original PSO algorithm (Eq. (2)), the users have to determine two acceleration coefficients, c_1 and c_2. Moreover, in order to strengthen the local search ability, [49] introduced an inertia weight parameter w, which is usually set to be in (0,1), into the velocity updating:

$$V_i^d \leftarrow w \times V_i^d + c_1 \times rand_i^d \times \left(pbest_i^d - X_i^d\right) + c_2 \times rand_i^d \times \left(gbest_i^d - X_i^d\right). \tag{4}$$

Compared with the original PSO, the updating of the velocity V_i^d in Eq. (4) is depending more heavily the previous experiences, i.e., *pbest* and *gbest*. It can be observed that there are usually three parameters required to be adapted beforehand, including inertia weight w, the acceleration coefficient and the number of particles.

In the past few years, many approaches have been applied to investigate how to adapt suitable parameters. In some early works, Shi et al. analyzed the impact of the inertia weight and maximum velocity in PSO [50]. They revealed that different combinations of inertia weights and maximum velocities may result in different performance on the Schaffer function. Based on the empirical analysis, it was concluded that a time-varying inertia weight can lead to better performance. This idea has been adopted by many works [5,32,56]. In [51], Shi et al. proposed a PSO variant, which implements a fuzzy rule to control the parameters. Following the same line of thinking, Parsopoulos et al. focused on the adaptation of the most important parameter of unified particle swarm optimization (UPSO) [40], which was previously proposed in [39]. According to the results of the experiments, the self-adaptive scheme proposed in [40] can well balance the capabilities of exploration and exploitation.

Due to the fact that most of the PSO research works related to adaptive schemes are empirical, Bergh et al. investigate the influence of parameters, such as the inertia term, from the perspective of particle trajectories for general swarms [5]. Experimental results on function optimization demonstrated that the performance of PSO is sensitive to the settings of the inertia weight and acceleration coefficient values. In order to make up this demerit, a heuristic for the initialization of the inertia weight and acceleration coefficient values was proposed in [5]. Besides the adaptation of parameters used for

Table 1
Procedure of PSO.

Original particle swarm optimization
Step (0) Initialization
Randomly initialize the positions of all particles $X = (X_1, X_2, \ldots, X_{ps})$ of size ps
Initialize the velocity $(V_1, V_2, \ldots, V_{ps})$
Set generation $t = 0$
Evaluate the fitness values $F = (f_1, f_2, \ldots, f_{ps})$ of X
Set X to be $pbest = (pbest_1, \ldots, pbest_{NP})$ for each particle
Set the particle with best fitness to be *gbest*
Step (1) Reproduction and updating loop
for $i = 1{:}ps$
Update the velocity v_i of particle x_i using Eq. (2)
$V_i^d \leftarrow V_i^d + c_1 \times rand_i^d \times (pbest_i^d - X_i^d) + c_2 \times rand_i^d \times (gbest_i^d - X_i^d)$
Update the position of particle x_i using Eq. (3)
$X_i^d \leftarrow X_i^d + V_i^d$
Evaluate the fitness value f_i of the new particle X_i
if X_i is better than $pbest_i$
Set X_i to be $pbest_i$
end if
if X_i is better than *gbest*
Set X_i to be *gbest*
end if
end for
Set $t = t + 1$
Step (2) If termination condition is not met, goto Step (1), otherwise end PSO

velocity updating, the self-adaptive setting of an extended parameter, the number of particles, has also gained much attention recently [10,19]. Benefitting from the particle size adaptation scheme, EPUS-PSO [19] shows much higher convergence speed on most problems compared with many state-of-the-art PSO variants.

We would like to emphasize that, compared with the above works of adapting the parameters, SLPSO mainly focuses on the self-adaptation of the probabilities of associated generating strategies. Although the parameter adaptation has been experimentally proven to be able to remarkably improve the performance of a generating strategy, the implementation of only one generating strategy cannot solve all problems even when the potential capability is fully explored, which can be concluded from the no free lunch theory [15]. In SLPSO, four velocity updating strategies with different strengths are used in parallel and adaptively applied in order to efficiently solve as many problems as possible. Inherently, in a certain sense, the parameter adaptation could still strengthen the sub-optimizers of the SLPSO framework.

2.2. How to obtain better exploration?

The original PSO has pretty good convergence ability, but also suffers the demerit of premature convergence [68], due to the loss of diversity [69]. Improving the exploration ability of PSO has been an active research topic in recent years.

In order to understand how PSO works on complex problems, Clerc et al. investigate its behavior from the perspective of the trajectory of an individual particle [12]. It is concluded that the suitable application of constriction coefficients can dynamically control the balance between exploration and exploitation. Also, a PSO variant with a constriction factor was proposed in [12] based on the analysis.

It has been investigated in [22,25,52] that the swarm topologies in PSO can remarkably influence the performance. Therefore, designing a suitable topology is another way to obtain better exploration. Mendes et al. claim that each individual should not only be simply influenced by the best performer among its neighbors [36]. Therefore, a fully informed particle swarm (FIPS) is proposed to utilize all the neighbors to update the velocity. According to the experimental results, FIPS shows excellent performance, especially on the Griewank function.

Liang et al. tried to preserve the diversity during the optimization procedure by utilizing a comprehensive learning strategy, which is the most important feature of CLPSO [28]. In CLPSO, *gbest* is not used when updating the particle velocity and moreover, the *pbest*s of all the other particles can be selected, based on the stochastic tournament model, to conduct the learning object to replace the *pbest* of the original PSO. Because of these improvements, CLPSO yields an at least ten times larger potential search region than the original PSO method during the optimization procedure on the Sphere and Rastrigin problems. Besides, the exploration of PSO has been improved by controlling the flight of particles recently [69,68].

An improved exploration is one of the most important objectives of SLPSO. In SLPSO, the velocity updating strategy of CLPSO is specially included as a sub-optimizer for complex multi-modal problems. At different optimization stages, the self-adaptive learning framework in SLPSO could give prominence to other generating strategies when the qualities of solutions generated by the CLPSO strategy decreases. In this case, the robustness of SLPSO is highly improved. The experimental results in the fifth section will validate that SLPSO outperforms CLPSO and FIPS, which have shown excellent performance on multi-modal problems, on almost all test problems.

2.3. How to incorporate other search techniques?

In order to strengthen the search ability, many researchers incorporate other search techniques into the PSO variants, such as cooperative co-evolution [4] and restart [31].

In [42], Potter and De Jong introduced cooperative co-evolution (CC) into EAs. The main idea of CC is to partition a complex problem into a number of sub-problems and then 'evolve' them, respectively, in multiple cycles. A cooperative co-evolution based PSO paradigm, named cooperative PSO (CPSO-H), is proposed in [4]. CPSO-H uses n swarms to separately search n dimensions and then, these n swarms are integrated by a global swarm. According to the experimental results, CPSO-H is able to effectively handle separable problems. Some other works also introduce co-evolution into PSO: Luo et al. implement a co-evolving framework to simultaneously optimize the problems and the control parameters. Refs. [17,32] adopt a competitive and cooperative co-evolutionary approach for PSO in multi-objective optimization for instance.

High convergence speed is a notable feature of the original PSO, but the risk of shrinking into the local optima is also very high. Therefore, restart [31] and collision-avoiding [6] strategies are often used to prevent the premature convergence of PSO. Besides, some common EA techniques are introduced into PSO to improve its performance. In [1], a selection operator is adopted to preserve the particles with best fitness values in the swarm. A new evolutionary PSO, which introduces the mutation operator into PSO, is proposed in [37]. Recently, many other techniques have been hybridized with PSO, such as ant colony optimization [20,21,48], memory strategy, scatter search [65] and harmony search [21].

In the above works, other search techniques were directly hybridized with PSO in a static way. For PSO itself, the potential search ability may not be explored fully. Therefore, to improve the performance of PSO on some specific aspects, we propose two sub-optimizers for SLPSO by implementing the main idea of other search techniques.

3. DbV, EbV and PSO-CL-pbest

In the first section, we stated that the performance of previous PSO variants on ill-conditioned problems is very poor [19,28,36,41]. In this section, we try to make up this demerit of PSO by proposing two new strategies, the difference-based velocity updating strategy (DbV) and the estimation-based velocity updating strategy (EbV). The main idea of these two strategies is to achieve an appropriate velocity by estimating the distribution of the swarm or the fitness landscape of the problem: in DbV, the velocity is completely recombined based on the difference information; EbV adopts an estimation of distribution algorithm (EDA) like strategy to effectively shrink the search space.

3.1. Difference-based velocity updating strategy

In most velocity updating strategies, such as the one described in Eq. (2), the generated velocity vector can be treated as the moderate modification of the old velocity vector according to the influences of neighborhood particles, which results in the fact that it is hard to be quickly adapted to the different optimization stages of ill-conditioned problems. Particularly on the Rosenbrock problem, most PSO variants have demonstrated very poor performance [19,28].

It has been testified in [43] that the differential evolution strategy, DE/current-to-rand/1 without crossover, is effective for rotated problems. The updating equation of DE/current-to-rand/1 is defined as follows:

$$U_i^d \leftarrow X_i^d + K \times \left(X_k^d - X_i^d\right) + F \times \left(X_j^d - X_h^d\right), \tag{5}$$

where X_j^d, X_j^d, X_k^d are individuals selected randomly from the population. It is observed that the above updating equation relies on the difference information only.

Motivated by the strategy DE/current-to-rand/1, DbV avoids progressively changing the velocity but completely recombines the velocity based on the difference information. Different to DE/current-to-rand/1, DbV not only utilizes the difference information to recombine the velocity vector. Instead, the *pbest* is used as an attractor to guide the flying direction of particle. In the DbV strategy, the following velocity updating equation is used:

$$Viad_i^d \leftarrow X_k^d - X_j^d, \tag{6}$$

$$c = N(0.5, 0.2).$$

$$V_i^d \leftarrow c \times Viad_i^d + c \times \left(pbest_i^d - X_i^d\right). \tag{7}$$

$$V_i^d = \min\left(V_{\max}^d, \max\left(V_i^d, V_{\min}^d\right)\right), \tag{8}$$

where X_k^d and X_j^d are the dth variables of two randomly selected other particles; $Viad_i^d$ is the difference vector; $N(0.5, 0.2)$ represents one number randomly generated according to the Gaussian distribution with mean 0.5 and standard deviation 0.2; $V_{\max}$ and $V_{\min}$ represent the pre-defined maximum and minimum velocity. The difference information is obtained by Eq. (6). Eq. (7) is used to generate the new velocity vector. It can be observed that this velocity updating does not rely on the past experience. Eq. (8) prevents this vector overstepping the pre-defined range.

3.2. Estimation-based velocity updating strategy

The high convergence speed is an important feature of the original PSO algorithm. However, the usage of the global elite "*gbest*" imposes a strong influence on the whole swarm [28], which easily misleads the search towards local optima. In this case, on complex multi-modal problems, the convergence speed of the original PSO algorithm is often very high at the beginning, but only lasts for a few generations. After that, the search will inevitably be trapped. Therefore, on such kind of problems, it is hard for self-adaptive learning mechanism to distinguish which is indeed the more effective strategy, because the qualities of the solutions generated by original PSO updating strategy are the best at the beginning. Thus, it would mislead the search towards local optima, which inhibits the advantages of other strategies on multi-modal problems. Under such circumstances, the original PSO updating strategy should not be included in SLPSO. Hence, we developed an alterative updating strategy which is able to achieve high convergence speed but avoids quickly being trapped by local optima on very complex multi-modal problems.

Estimation of distribution algorithms (EDAs) are modelling-based optimization algorithms [26,27]. Compared with the evolutionary operators of genetic algorithm (GA), evolution strategy (ES) and evolutionary programming (EP), that mainly operate on the *members* of the population, at each generation, the updating operators of EDAs manipulate the distribution of the whole *population* itself by learning a probability model describing the distribution of its individuals. EDAs have shown much higher convergence speed than GA, DE and PSO on unimodal problems, as pointed out in [61,62,66]. In [61], a mixed probability model of Cauchy and Gaussian distributions is proposed to get a proper trade-off between population diversity and convergence speed. Inspired by these works, we propose a new velocity updating strategy based on the local estimation of distribution of three positions, which is defined as follows:

$$c = \frac{(D-1)N(0,1)}{D} + \frac{C(0,1)}{D}. \tag{9}$$

$$V_i^d \leftarrow \left(mean_i^d - X_i^d\right) + \frac{c}{\sqrt{3}}\sqrt{\left(pbest_i^d - mean_i^d\right)^2 + \left(X_i^d - mean_i^d\right)^2 + \left(X_k^d - mean_i^d\right)^2}. \tag{10}$$

$$V_i^d = \min\left(V_{\max}^d, \max\left(V_i^d, V_{\min}^d\right)\right), \tag{11}$$

where X_k^d is the dth component of a randomly selected particle; *mean* is the mean of best 20% particles' positions; c is a coefficient in Eq. (10) that determines the influence of mean shifts of $X(i)$, its *pbest* and $X(k)$ randomly selected from swarm. From Eq. (9), it can be observed that c is randomly sampled from a mixed Gaussian and Cauchy distribution, represented by $N(0,1)$ and $C(0,1)$, respectively. Different from EDAs, EbV does not require the building of a precise probabilistic model of the top individuals, but just computes a jumping vector by roughly and locally estimating the particle distribution around particle i. On problems with simple fitness landscapes, EbV can provide a very high convergence speed. It has been empirically proven in [61] that the introduction of the Cauchy distribution can effective guarantee the diversity on complex multi-modal problems, although the quality of these solutions may be very poor on these problems. Therefore, in this paper, we adopt EbV as a alternative velocity updating strategy to provide the necessary convergence ability.

3.3. CLPSO and PSO-CL-pbest

The comprehensive learning strategy proposed in [28] is effective to improve the performance of PSO on multi-modal problems. In comprehensive learning PSO (CLPSO), the velocity of each component of each particle has a chance to be influenced by the *pbest* of every other particle. Such a strategy encourages the diversity of the swarm, and is therefore able to solve multi-modal problems. In CLPSO, the velocity is updated as follows [28]:

$$V_i^d \leftarrow w \times V_i^d + c \times rand_i^d \times \left(pbest_{f_i(d)}^d - X_i^d\right). \tag{12}$$

$$V_i^d = \min\left(V_{\max}^d, \max\left(V_i^d, V_{\min}^d\right)\right), \tag{13}$$

where $f_i = [f_i(1), f_i(2), \ldots, f_i(D)]$ denotes the D particles selected to update the variables of ith particle. It is possible that $f_i(j) = i$.

It has been investigated in [28] that the search range of CLPSO is very large on almost all problems. Therefore, we develop a new variant that searches a relatively smaller region, named PSO-CL-pbest. The velocity updating strategy is defined as follows:

$$V_i^d \leftarrow w \times V_i^d + 0.5 \times c \times rand_i \times \left(pbest_{f_i(d)}^d - X_i^d + pbest_i^d - X_i^d\right). \tag{14}$$

$$V_i^d = \min\left(V_{\max}^d, \max\left(V_i^d, V_{\min}^d\right)\right). \tag{15}$$

Compared with CLPSO, PSO-CL-pbest uses $rand_i$ instead of $rand_i^d$, which implies that a fixed random number is used to update all components of one velocity vector. As discussed in [28], $rand_i$ can result in smaller search range than $rand_i^d$. Besides, the $pbest_i^d$ is involved to attract the search towards its own *pbest*, which is also beneficial to shrinking the search region.

4. Self-adaptive learning based particle swarm optimization

The purpose of designing a self-adaptive learning PSO framework is to improve the universality or robustness of PSO to diverse problems with various features. The basic idea is to combine multiple effective velocity updating strategies and to steer them in a parallel and adaptive way. In more details, for different problems and at different stages of the optimization process, multiple strategies may be assigned a different execution probability based on the feedback of solution quality they produced previously.

Before presenting the SLPSO algorithm, we want to list again the properties of the four velocity updating strategies, CLPSO, PSO-CL-pbest, DbV, and EbV, that will be adopted in SLPSO:

- CLPSO has demonstrated pretty good exploration ability. It is much suitable for handling multi-modal problems, but has a low convergence speed for uni-modal problems.
- Compared with CLPSO, PSO-CL-pbest has better exploitation ability but a little worse exploration ability. It can solve many multi-modal problems with much higher convergence speed than CLPSO.
- The essential of DbV is to make use of the difference information to effectively adapt the velocity for the rotated problems. Besides, it is also beneficial to solve uni-modal problems and relatively simple multi-modal problems.
- EbV is specially designed to yield fast convergence for uni-model problems.

Generally speaking, the criteria of choosing these four updating strategies are that they have distinct properties and strengths that cover diverse aspects.

4.1. Self-adaptive learning strategy

In principle, the ultimate purpose of using a self-adaptive learning strategy is to increase the probabilities of using suitable updating strategies for different problems, based on the feedback of their previous performance. It is apparent that this is crucial to extend the applicability of SLPSO. Historically, the idea of hybridizing multiple low-level heuristics to form a high-level heuristic method has gained much attention [7]. Successful examples were observed on combinatorial optimization problems [7,13,14], global numerical optimization problems [33,43,46,54,59], and multi-objective optimization problems [58]. Several strategies [7,43,59] have been proposed to tune the contribution of various sub-algorithms adaptively according to their previous performance, and have shown pretty good effect. In SLPSO, the fundamental idea of self-adaptive learning based framework is similar to those used in [43,59], but the implementation is different. A learning rate α is used in SLPSO to control the learning speed of the execution probabilities during the optimization procedure, which is similar to the learning mechanism used in population-based incremental learning algorithm (PBIL) [47]. With such gradual learning mechanism, SLPSO can reduce the impact of fitness assignment of different landscapes to the learning mechanism.

In SLPSO, for each strategy, we assign an execution probability $proSTR_i$ to determine the probability that the ith strategy be adopted to update each particle. Initially, we assign equal probabilities for all strategies, that is $proSTR_i = 0.25$, for $i = 1,\ldots,4$, and set accumulators of four strategies $S_i = 0$. At each generation, the particles are sorted based on their fitness values. Then, each particle is assigned a weight, i.e., $w_j = \frac{\log(ps-j+1)}{\log(1)+\cdots+\log(ps)}$ for the jth best particle, $j = 1,\ldots,ps$. Finally, the weights are added to the accumulators of their associated updating strategies. After a fixed number of generations Gs, the following rule is used to update the execution probability of jth updating strategy:

$$proSTR'_j = (1-\alpha)proSTR_j + \alpha\frac{S_j}{Gs}, \tag{16}$$

$$proSTR_j = proSTR'_j/\left(proSTR'_1 + \cdots + proSTR'_4\right), \tag{17}$$

where $proSTR'_j$ is the temporal execution probability; α is the learning coefficient, which is used to control the updating proportion. S_j is divided by Gs to assure that it is smaller than 1. Eq. (17) normalizes the execution probabilities. It can be expected that the strategies which have generated higher-quality solutions tend to increase their execution probabilities step by step. The details of SLPSO are shown in Table 2.

4.2. Search range analysis

In order to analyze the search behavior of SLPSO, we compare its potential search region with that of CLPSO and PSO. A search range comparison between PSO and CLPSO has been carried out in [28]. The potential search range of PSO for the ith particle can be obtained by [28]:

$$r^d_{1i} = L^d_{1i} + L^d_{2i} = \left|gbest^d - X^d_i\right| + \left|pbest^d_i - X^d_i\right|, \tag{18}$$

where the search range of ith particle is composed of two independent parts. Similarly, the potential search range of CLPSO for the ith particle is defined as follows [28]:

$$r^d_{2i} = \max\left(pbest^d_j, X^d_i\right) - \min\left(pbest^d_j, X^d_i\right), \tag{19}$$

where $j = 1,2,\ldots,ps$. The potential search range of DbV for the ith particle can be obtained by:

$$r^d_{3i} = \max\left(X^d_j\right) - \min\left(X^d_j\right) - \max\left(pbest^d_j, X^d_i\right) - \min\left(pbest^d_j - X^d_i\right), \tag{20}$$

where $j = 1,2,\ldots,ps$. The potential search range of EbV for the ith particle can be obtained by:

$$r^d_{4i} = \max\left(\sqrt{\left(pbest^d_i - mean^d_i\right)^2 + \left(X^d_i - mean^d_i\right)^2 + \left(X^d_j - mean^d_i\right)^2}\right) - \min\left(\sqrt{\left(pbest^d_i - mean^d_i\right)^2 + \left(X^d_i - mean^d_i\right)^2 + \left(X^d_j - mean^d_i\right)^2}\right). \tag{21}$$

Therefore, the volumes of these four updating strategies potential search space for the ith particle are $R_{1i} = \prod_i^D r_{1i} \ldots R_{4i} = \prod_i^D r_{4i}$. For SLPSO, the potential search range of separate particle should be calculated using its associated equation of the updating strategy.

Two typical problems, the uni-modal Sphere problem and the multi-modal Rastrigin problem (function 1 and 9 in Table 4), are selected to conduct this investigation. All three algorithms perform 30 independent runs, and the mean search range change curves are depicted in Fig. 1. It can be observed that on the Sphere problem, the speed of the shrinking of the potential search range of SLPSO is comparable with PSO after a learning period, while that of CLPSO keeps slow throughout the optimization process. As to the Rastrigin problem, which contains many deep local optima, it can be observed in Fig. 1 that

Table 2
Procedure of SLPSO.

Self-adaptive learning based particle swarm optimization

Step (0) Initialization
Randomly initialize the positions of all particles $X = (X_1, X_2, \ldots, X_{ps})$ of size ps
Initialize the velocity $(V_1, V_2, \ldots, V_{ps})$
Set generation $t = 0$
Evaluate the fitness values $F = (fit_1, fit_2, \ldots, fit_{ps})$ of X
Set X to be $pbest = (pbest_1, \ldots, pbest_{ps})$ for each particle
Set the particle with best fitness to be $gbest$
Set probabilities of using 4 strategies $proSTR_i = 0.25$, $i = 1, \ldots, 4$
Set the learning period $Gs = 10$, learning coefficient $\alpha = 1/6$
Set accumulators of 4 strategies $S_i = 0$, $i = 1, \ldots, 4$
Set weight $w_i = \frac{log(ps-i+1)}{log(1)+\cdots+log(ps)}$, $i = 1, \ldots, ps$

Step (1) Reproduction and updating loop
for $i = 1 : ps$
  Select the jth strategy by roulette wheel selection based on $proSTR$ for the ith particle
  Update the velocity v_i and position x_i of particle x_i using the jth strategy
  Evaluate the fitness value fit_i of the new particle X_i
  if X_i is better than $pbest_i$
    Set X_i to be $pbest_i$
  end if
  if X_i is better than $gbest$
    Set X_i to be $gbest$
  end if
end for
Sort the fitness value F in descending order $F' = (fit'_1, fit'_2, \ldots, fit'_{NP})$
for $i = 1 : ps$
  Find out the jth strategy that generates the particle with fit'_i
  $S_j = S_j + w_i$
end for
if $t\%Gs == 0$
  for $j = 1{:}4$
    $proSTR'_j = (1 - \alpha)proSTR_j + \alpha S_j/Gs$
    $S_j = 0$
  end for
  for $j = 1 : 4$
    $proSTR_j = proSTR'_j/(proSTR'_1 + \cdots + proSTR'_4)$
  end for
end if
Set $t = t + 1$

Step (2) If termination condition is not met, goto Step (1), otherwise end PSO

the potential range of PSO quickly shrinks, which implies that it has poor exploitation ability. When comparing SLPSO and CLPSO, CLPSO's potential search range is slightly larger than SLPSO's when it is faced with the difficulty of many local optima, while the duration of SLPSO, measured by Fitness Evaluation Size (FES), lasts a little longer than CLPSO. However, when the basin that contains the global optimum is found, SLPSO can quickly shrink its search space. In summary, it can be concluded that SLPSO can definitely self-adaptively and effectively control the shrinking speed of potential search range according to the landscape feature of problem.

5. Experimental study on function optimization

5.1. Algorithms for comparison

In order to evaluate the effectiveness and efficiency of SLPSO, we compare its performance with those of eight state-of-the-art PSO variants. The algorithms for comparison are listed as follows:

- PSO-w: PSO with inertia weight [49];
- PSO-cf: PSO with constriction factor [12];
- PSO-cf-local: local version of PSO with constriction factor [25];
- FIPS-PSO: fully informed PSO [36];
- FDR-PSO: Fitness-distance-ratio based PSO [41];
- CPSO-H: cooperative based PSO [4];
- CLPSO: comprehensive learning PSO [28].

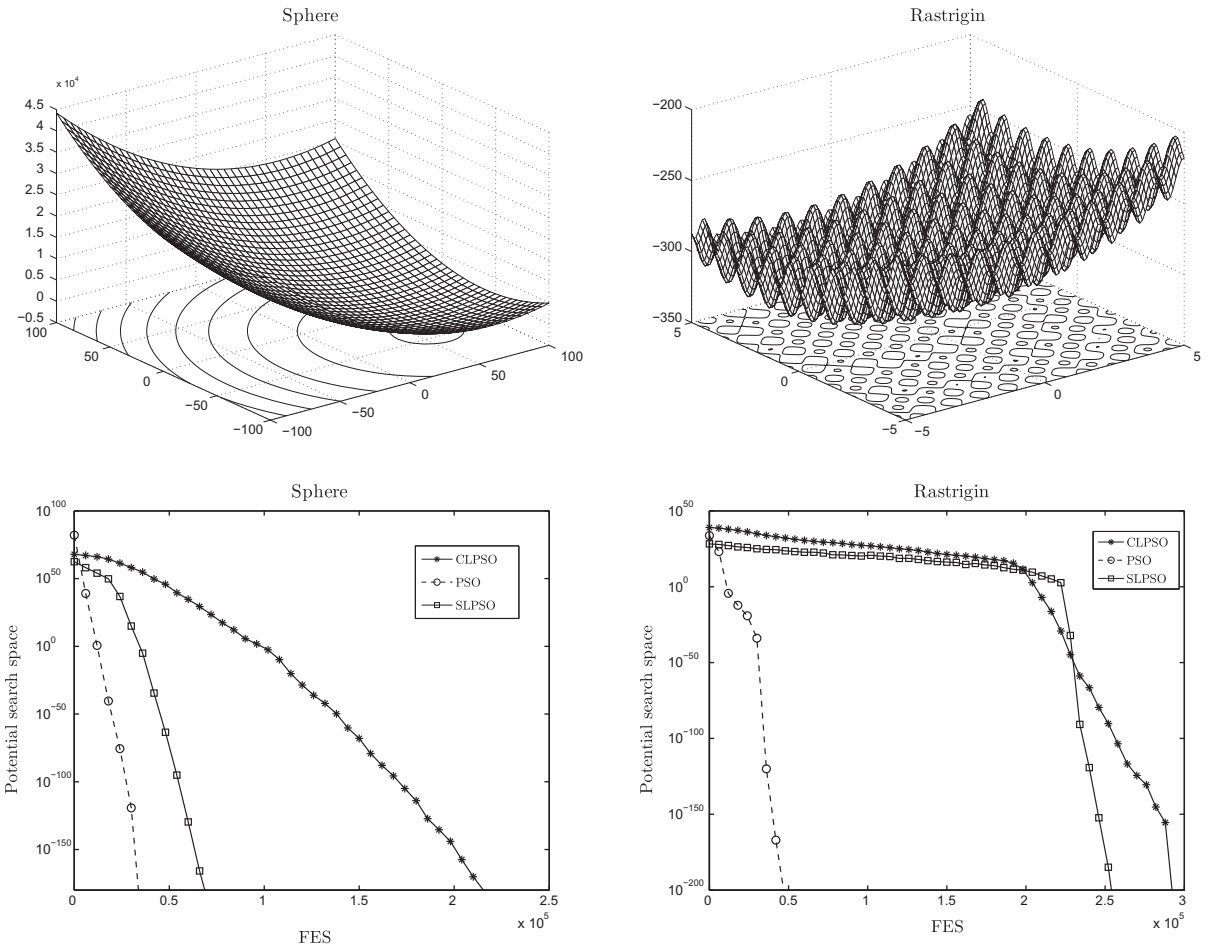

Fig. 1. Comparison of the potential search space among three algorithms.

In general, these algorithms have different strengthes. Amongst these algorithms, CLPSO has exhibited the top capability of handling multi-modal problems [28], while PSO-cf performs well on uni-modal problems [12,28]. CPSO-H, using one-dimensional decomposition has shown excellent performance on separable problems [4]. The parameter settings for all algorithms are inherited from the referenced papers. Some important parameters' settings are summarized in Table 3.

5.2. Test functions

To provide a comprehensive comparison, 26 numerical optimization problems with different characteristics, such as uni-modality, multi-modality, rotation, ill-condition, mis-scale, and noise, are used to conduct this experiment. As discussed in [29,43,53], there exist two major problems in the test functions used in previous literatures [64]: (1) the global optimum is

Table 3
Parameters setting of the compared algorithms.

	Inertia weight	Acceleration coefficient
PSO-w	$w(g) = 0.9 - \frac{0.5 * g}{\max_g en}$	$f_1 = f_2 = 1.49$
PSO-cf	$w = 0.729$	$c_1 = c_2 = 1.49445$
PSO-cf-local	$w = 0.729$	$c_1 = c_2 = 1.49445$
FIPS-PSO	$w = 0.729$	$c_1 = c_2 = 2.0$
FDR-PSO	$w(g) = 0.9 - \frac{0.5*g}{\max_g en}$	$f_1 = f_2 = 1.0, f_3 = 2.0$
CPSO-H	$w(g) = 0.9 - \frac{0.5*g}{\max_g en}$	$c_1 = c_2 = 2.0$
CLPSO	$w(g) = 0.9 - \frac{0.5*g}{\max_g en}$	$c_1 = c_2 = 1.49445$

always located at the center of the search space; (2) the distribution of local optima is regular and the variables are separable. In order to better approximate real-world problem behavior and to make the benchmark functions more challenging, the classical test functions are shifted and rotated with a certain scale.

In Table 4, the global optima of all functions are shifted to different numerical values for different dimensions ($z = x - o$), where o is the location of the new global optimum. These test functions can be divided into three groups: (1) test functions with separable variables; (2) test functions with mis-scaled variables; (3) test functions with noisy landscapes. For the second set of problems, the classical test functions are rotated by $z = M(x - o)$, where M is an orthogonal rotation matrix, to avoid local optima lying along the coordinate axes while retaining the properties of the test functions. The rotated functions are summarized in Table 5.

Table 4
Classical benchmark problems to be minimized.

Num	S	R	Problems	Objective function
Group 1: Classical test functions (non-linear, separable, scalable) ($z = x - o$)				
fun1	√		**Shifted Sphere**	$f_1(x) = \sum_{i=1}^{n} z_i^2 + f_{bias}$
fun2	√		**Shifted Schwefel 1.2**	$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j^2\right) + f_{bias}$
fun3	√		**Shifted Schwefel 2.22**	$f_3(x) = \sum_{i=1}^{n} \lvert z_i\rvert + \prod_{i=1}^{n} \lvert z_i\rvert + f_{bias}$
fun4	√		**Shifted Schwefel 2.21**	$f_4(x) = \max\lvert z_i\rvert + f_{bias}$
fun5	√		**Shifted Rochenbrock**	$f_5(x) = \sum_{i=1}^{D} \left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right) + f_{bias}$
fun6	√		**Shifted Ackley**	$f_6(x) = -20 \cdot exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) + e - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)\right) + 20 + f_{bias}$
fun7	√		**Shifted Griewank**	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos\left(\frac{z_i}{\sqrt{i}}\right) + f_{bias}$
fun8	√		**Shifted Rastrigin**	$f_8(x) = \sum_{i=1}^{n} (z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun9	√		**Noncontinues Rastrigin**	$f_9(x) = \sum_{i=1}^{n} (y_i^2) - 10cos(2\pi y_i) + 10$, $y_{i+1} = \begin{cases} z_i, if \ \lvert z_i\rvert < 1/2 \\ round(2 * z_i)/2, otherwise, \end{cases}$
fun10	√		**Shifted Penalized 1**	$f_{10} = \frac{\pi}{30}\left\{10\sin^2 + \sum_{i=1}^{2} 9(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1} + (y_n - 1)^2)] + \sum_{i=1}^{30} u(x_i, 10, 100, 4)\right\}$, u and y are shown in Appendix
fun11	√		**Shifted Penalized 2**	$f_{11} = 0.1\left\{\sin^2(\pi 3x_1) + \sum_{i=1}^{2} 9(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2\right\} \cdot [1 + \sin^2(2\pi x_3 0)] + \sum_{i=1}^{30} u(x_i, 5, 100, 4)\}$, u and y are shown in Appendix
Group 2: Test functions that are mis-scaled ($z = x - o$)				
fun12	√		**Shifted Rochenbrock100**	$f_{12}(x) = \sum_{i=1}^{D} (100((a_i z_i)^2 - (a_{i+1} z_{i+1}))^2 + ((a_i z_i) - 1)^2) + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun13	√		**Shifted Rastrigin10**	$f_{13}(x) = \sum_{i=1}^{n} ((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun14	√		**Shifted Rastrigin1000**	$f_{14}(x) = \sum_{i=1}^{n} ((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}$, $a_i = 1000^{\frac{i-1}{D-1}}$
Group 3: Test function with noise ($z = x - o$)				
fun15	√		**Noise Schwefel 1.2**	$f_{15}(x) = \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j^2\right)\right)(1 + 0.4\lvert N(0,1)\rvert) + f_{bias}$

Table 5
Rotated benchmark problems to be minimized.

Num	S	R	Problems	Objective function
Group 4: Rotated test functions (non-linear, non-separable, scalable) ($z = M(x - o)$)				
fun16	√	√	**Rotated Sphere**	$f_{16}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$
fun17	√	√	**Rotated Tablet**	$f_{17}(x) = (1000x_1)^2 + \sum_{i=2}^{D} z_i^2 + f_{bias}$
fun18	√	√	**Rotated Ellipse**	$f_{18}(x) = \sum_{i=1}^{D} (20^{\frac{i-1}{D-1}} z_i)^2 + f_{bias}$
fun19	√	√	**Rotated diff pow**	$f_{19}(x) = \sum_{i=1}^{D} z_i^{2+a_i} + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun20	√	√	**Rotated Schwefel 2.21**	$f_{20}(x) = \max\lvert z_i\rvert + f_{bias}$
fun21	√	√	**Rotated Rochenbrock**	$f_{21}(x) = \sum_{i=1}^{D} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$
fun22	√	√	**Rotated Ackley**	$f_{22}(x) = -20 \cdot exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) + e - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)\right) + 20 + f_{bias}$
fun23	√	√	**Rotated Griewank**	$f_{23}(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos\left(\frac{z_i}{\sqrt{i}}\right) + f_{bias}$
fun24	√	√	**Rotated Rastrigin**	$f_{24}(x) = \sum_{i=1}^{n} (z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun25	√	√	**Noise Schwefel 1.2**	$f_{25}(x) = \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j^2\right)\right)(1 + 0.4\lvert N(0,1)\rvert) + f_{bias}$
fun26	√	√	**Noise Quadric**	$f_{26}(x) = \sum_{i=1}^{n} i z_i^4 + random[0,1) + f_{bias}$

5.3. Experimental results

In this experiment, we set the dimension size of all test functions to be 30, set the maximum number of function evaluations to be 300,000, and set the number of particles to be 50. For all test functions, the algorithms carry out 30 independent runs. In Tables 6–8, the mean, standard deviation, and the number of successful runs are recorded. A run is considered to be successful if at least one solution was discovered during its course whose fitness value is not worse than $(fit(x^*) + 1e - 5)$. Moreover, when all 30 runs are successful, the average number of function evaluations (NFE) required to find the global optima are also recorded. The best results are typed in bold. The ranking of the algorithms is based on the successful performance *SP* [3], which is defined as follows:

$$SP = mean(\text{function evaluations of successful runs}) \times \frac{\text{all runs (30)}}{\text{number of successful runs}}. \tag{22}$$

Figs. 2–5 depict the convergence characteristics in terms of the best fitness value of the median run of each algorithm.

Table 6
Optimization results for functions 1–10.

Algorithm	Mean	Standard deviation	NFE	Sruns	Mean	Standard deviation	NFE	Sruns
Function 1, 2	Shifted Sphere				Shifted Schwefel 1.2			
SLPSO	0.00E+00	0.00E+00	43,980	30	**7.69E−13**	**4.27E−13**	**149,872**	**30**
CLPSO	0.00E+00	0.00E+00	122,161	30	1.16E+03	2.44E+02		0
PSO-w	0.00E+00	0.00E+00	193,017	30	1.11E+00	1.16E+00		0
wFIPS-PSO	0.00E+00	0.00E+00	146,197	30	2.08E+02	8.98E+01		0
FDR-PSO	0.00E+00	0.00E+00	92,165	30	5.58E−12	7.57E−12	215,618	30
PSO-cf-local	0.00E+00	0.00E+00	40,295	30	1.53E−04	1.68E−04		1
PSO-cf	**0.00E+00**	**0.00E+00**	**20,333**	**30**	1.29E−12	3.88E−13	151,095	30
PSO-CL-pbest	0.00E+00	0.00E+00	10,9731	30	1.31E+03	2.26E+02		0
CPSO-H	1.47E−08	1.49E−08	149,044	30	2.79E+03	5.98E+03		0
Function 3, 4	Shifted Schwefel 2.22				Shifted Schwefel 2.21			
SLPSO	0.00E+00	0.00E+00	55,859	30	5.17E−05	1.28E−04		13
CLPSO	0.00E+00	0.00E+00	152,608	30	2.11E+00	2.45E−01		0
PSO-w	0.00E+00	0.00E+00	194,915	30	5.35E−01	3.39E−01		0
wFIPS-PSO	1.44E−09	4.19E−10	190,198	30	1.25E−02	4.34E−03		0
FDR-PSO	0.00E+00	0.00E+00	122,498	30	1.84E−04	2.17E−04		1
PSO-cf-local	0.00E+00	0.00E+00	51,544	30	2.68E−06	3.85E−06		28
PSO-cf	**4.71E−11**	**1.80E−10**	**31,691**	**30**	**9.51E−10**	**1.07E−09**	**177,887**	**30**
PSO-CL-pbest	0.00E+00	0.00E+00	135,342	30	1.15E+00	2.67E−01		0
CPSO-H	2.86E−05	1.36E−05		1	5.14E−03	4.49E−03		0
Function 5, 6	Shifted Rosenbrock				Shifted Ackley			
SLPSO	**2.66E−01**	**1.01E+00**		**28**	**0.00E+00**	**0.00E+00**	**51,583**	**30**
CLPSO	4.95E+00	3.79E+00		0	7.77E−13	1.49E−13	166,425	30
PSO-w	6.46E+01	8.01E+01		0	4.76E−13	4.83E−14	211,209	30
wFIPS-PSO	2.52E+01	9.08E−01		0	1.39E−08	2.98E−09	206,600	30
FDR-PSO	4.63E+00	1.37E+01		0	0.00E+00	0.00E+00	129,273	30
PSO-cf-local	2.98E+01	4.07E+01		0	0.00E+00	0.00E+00	56,976	30
PSO-cf	8.88E+00	1.46E+01		0	8.51E−01	1.01E+00		15
PSO-CL-pbest	2.40E+00	2.08E+00		0	9.34E−13	2.45E−13	163,236	30
CPSO-H	2.77E+01	2.86E+01		0	2.44E−05	1.35E−05		1
Function 7, 8	Shifted Griewank				Shifted Rastrigin			
SLPSO	1.81E−03	4.79E−03		26	0.00E+00	0.00E+00	196,749	30
CLPSO	0.00E+00	0.00E+00	151,708	30	**0.00E+00**	**0.00E+00**	**195,815**	**30**
PSO-w	8.73E−02	1.18E−01		2	1.75E+01	5.97E+00		0
wFIPS-PSO	2.72E−07	1.18E−06	183,531	30	6.39E+01	1.12E+01		0
FDR-PSO	1.53E−02	1.42E−02		5	3.17E+01	6.95E+00		0
PSO-cf-local	5.34E−03	7.46E−03		17	4.10E+01	1.13E+01		0
PSO-cf	1.71E−02	1.78E−02		8	6.45E+01	2.05E+01		0
PSO-CL-pbest	**0.00E+00**	**0.00E+00**	**147,934**	**30**	5.04E+00	1.36E+00		0
CPSO-H	1.20E−01	2.18E−01		4	3.32E−02	1.82E−01		29
Function 9, 10	Noncontinues Rastrigin				Shifted Penalized 1			
SLPSO	0.00E+00	0.00E+00	209,437	30	**0.00E+00**	**0.00E+00**	**35,878**	**30**
CLPSO	0.00E+00	0.00E+00	204,993	30	0.00E+00	0.00E+00	120,170	30
PSO-w	6.63E+00	5.09E+00		0	1.38E−02	3.58E−02		26
wFIPS-PSO	6.19E+01	1.16E+01		0	0.00E+00	0.00E+00	122,361	30
FDR-PSO	1.24E+01	7.45E+00		0	0.00E+00	0.00E+00	105,748	30
PSO-cf-local	1.08E+01	8.95E+00		1	3.11E−02	1.16E−01		26
PSO-cf	3.25E+01	2.08E+01		0	1.32E−01	3.45E−01		21
PSO-CL-pbest	8.14E+00	1.41E+00		0	0.00E+00	0.00E+00	99,144	30
CPSO-H	**3.51E−09**	**3.84E−09**	**116,133**	**30**	8.48E−11	1.41E−10	52,866	30

Table 7
Optimization results for functions 11–20.

Algorithm	Mean	Standard deviation	NFE	Sruns	Mean	Standard deviation	NFE	Sruns
Function 11, 12	Shifted Penalized 2				Shifted Rosenbrock100			
SLPSO	**0.00E+00**	**0.00E+00**	**38,761**	**30**	**7.88E+02**	**2.56E+03**		**17**
CLPSO	0.00E+00	0.00E+00	130,177	30	1.09E+03	3.45E+03		0
PSO-w	1.10E−03	3.35E−03		27	2.18E+05	8.24E+05		0
wFIPS-PSO	0.00E+00	0.00E+00	140,728	30	7.37E+04	3.14E+05		0
FDR-PSO	7.32E−04	2.79E−03		28	1.25E+05	4.14E+05		0
PSO-cf-local	3.66E−04	2.01E−03		29	9.11E+04	3.72E+05		0
PSO-cf	1.06E−02	2.09E−02		16	2.57E+04	4.91E+04		0
PSO-CL-pbest	0.00E+00	0.00E+00	111,279	30	5.62E+02	8.53E+02		0
CPSO-H	3.92E−09	1.15E−08	95,309	30	3.71E+06	4.38E+06		0
Function 13, 14	Shifted Rastrigin10				Shifted Rastrigin100			
SLPSO	0.00E+00	0.00E+00	225,475	30	0.00E+00	0.00E+00	234,253	30
CLPSO	0.00E+00	0.00E+00	218,073	30	**0.00E+00**	**0.00E+00**	**226,863**	**30**
PSO-w	2.53E+01	6.59E+00		0	3.05E+01	7.97E+00		0
wFIPS-PSO	7.17E+01	9.23E+00		0	8.59E+01	1.38E+01		0
FDR-PSO	2.93E+01	1.03E+01		0	5.31E+01	1.25E+01		0
PSO-cf-local	4.88E+01	1.35E+01		0	6.20E+01	3.87E+01		0
PSO-cf	9.30E+01	2.85E+01		0	1.58E+02	1.20E+02		0
PSO-CL-pbest	3.00E+00	1.06E+00		0	3.76E+00	1.06E+00		0
CPSO-H	**1.23E−07**	**1.86E−07**	**186,415**	**30**	6.14E−04	1.27E−03		1
Function 15, 16	Noise Schewefel 1.2				Rotated Sphere			
SLPSO	**2.32E−02**	**8.92E−02**		**0**	0.00E+00	0.00E+00	49,346	30
CLPSO	7.25E+03	1.37E+03		0	4.21E−10	6.18E−10	190,125	30
PSO-w	4.68E+02	3.14E+02		0	0.00E+00	4.56E−14	201,639	30
wFIPS-PSO	1.52E+03	5.44E+02		0	7.54E−13	3.26E−13	169,215	30
FDR-PSO	4.65E+02	3.66E+02		0	0.00E+00	0.00E+00	99,060	30
PSO-cf-local	5.71E+02	4.59E+02		0	0.00E+00	0.00E+00	47,289	30
PSO-cf	1.99E+02	2.89E+02		0	**0.00E+00**	**0.00E+00**	**24,010**	**30**
PSO-CL-pbest	7.73E+03	1.29E+03		0	2.63E−05	2.00E−05		8
CPSO-H	2.44E+04	8.49E+03		0	8.10E−08	1.02E−07	185,832	30
Function 17, 18	Rotated Tablet				Rotated Ellipse			
SLPSO	**1.66E−04**	**3.59E−04**		**16**	**2.22E−12**	**7.06E−13**	**146,787**	**30**
CLPSO	4.55E+03	1.11E+03		0	4.07E+03	9.37E+02		0
PSO-w	1.28E+02	7.61E+01		0	1.15E+02	1.49E+02		0
wFIPS-PSO	8.45E+02	2.14E+02		0	1.51E+03	7.14E+02		0
FDR-PSO	1.22E−01	1.47E−01		0	8.44E−07	2.00E−06	237,028	30
PSO-cf-local	4.50E+00	3.03E+00		0	7.66E−01	1.09E+00		0
PSO-cf	1.67E−04	2.38E−04		5	3.46E−03	1.13E−02		4
PSO-CL-pbest	6.12E+03	1.16E+03		0	4.46E+03	1.43E+03		0
CPSO-H	1.65E+05	4.61E+04		0	7.63E+03	6.69E+03		0
Function 19, 20	Rotated diff pow				Rotated Schwefel 2.21			
SLPSO	**1.97E−08**	**1.40E−08**	**147,666**	**30**	**4.51E−11**	**1.16E−10**	**81,481**	**30**
CLPSO	8.02E+08	9.22E+08		0	9.71E−01	2.38E−01		0
PSO-w	1.48E+10	3.26E+10		0	2.50E−01	3.03E−01		0
wFIPS-PSO	4.54E+10	9.01E+10		0	1.36E−04	4.89E−05		0
FDR-PSO	2.88E−06	3.14E−06		29	1.78E−02	4.21E−02		6
PSO-cf-local	6.62E+04	9.60E+04		0	7.98E−02	2.03E−01		14
PSO-cf	2.51E+05	4.03E+05		1	4.11E−02	1.40E−01		17
PSO-CL-pbest	1.23E+07	8.28E+06		0	2.87E−01	1.03E−01		0
CPSO-H	1.66E+10	4.71E+10		0	5.43E+01	7.52E+00		0

5.4. Discussion

5.4.1. Experimental results on group 1

It has been presented above that most of test functions in group 1 of Table 4 are separable and hence, can be solved by divided-and-conquer methods, such as [4,63]. It is observed in Tables 6 and 7 that the successful rates of all algorithms can reach 100% only on the Sphere problem. Actually, only SLPSO is able to provide at least one successful run for all problems, while the other algorithms fail on three (PSO-cf-local) or more problems. Therefore, it is evident that SLPSO shows the best universality on these problems.

The functions 1–5 are uni-modal problems. Amongst these five problems, function 1 and 3 are relatively easy ones, because almost all algorithms can solve them with 100% successful rate, see Table 6. For these two easy problems, the NFEs of PSO-cf are the lowest and followed by PSO-cf-local and SLPSO, which means that it provides the highest convergence speed on functions 1 and 3. This is due to the fact that the PSO-cf is specifically designed to fully make use of the "*gbest*". SLPSO

Table 8
Optimization results for functions 21–26.

Algorithm	Mean	Standard deviation	NFE	Sruns	Mean	Standard deviation	NFE	Sruns
Function 21, 22	Rotated Rosenbrock				Rotated Ackley			
SLPSO	**1.20E+02**	**3.81E+02**		**0**	**0.00E+00**	**0.00E+00**	**52,575**	**30**
CLPSO	2.56E+02	3.04E+02		0	1.16E−02	9.10E−03		0
PSO-w	4.62E+05	7.88E+05		0	2.34E+00	7.59E−01		1
wFIPS-PSO	2.89E+01	4.15E+00		0	2.24E−08	5.60E−09	213,274	30
FDR-PSO	7.92E+02	2.19E+03		0	1.19E+00	9.79E−01		10
PSO-cf-local	6.16E+02	1.93E+03		0	2.40E−01	4.98E−01		24
PSO-cf	1.18E+03	2.98E+03		0	1.95E+00	9.55E−01		4
PSO-CL-pbest	2.03E+02	3.61E+02		0	2.09E−02	2.48E−02		0
CPSO-H	1.62E+03	3.78E+03		0	1.76E+01	3.96E+00		0
Function 23, 24	Rotated Griewank				Rotated Rastrigin			
SLPSO	**5.34E−03**	**6.82E−03**		**17**	**3.15E+01**	**6.41E+00**		**0**
CLPSO	3.36E−02	2.00E−02		0	1.03E+02	1.26E+01		0
PSO-w	2.07E−01	3.84E−01		0	8.29E+01	5.00E+01		0
wFIPS-PSO	1.14E−03	3.00E−03		8	1.75E+02	8.79E+00		0
FDR-PSO	8.53E−03	9.94E−03		12	5.60E+01	1.80E+01		0
PSO-cf-local	1.04E−02	1.03E−02		9	5.26E+01	1.43E+01		0
PSO-cf	9.85E−03	7.99E−03		8	1.09E+02	3.63E+01		0
PSO-CL-pbest	2.68E−01	5.67E−02		0	1.22E+02	1.56E+01		0
CPSO-H	1.66E+00	2.10E−01		0	3.77E+02	1.10E+02		0
Function 25, 26	Rotated Noise Schwefel 1.2				Rotated Noise Quadric			
SLPSO	**7.79E−04**	**2.24E−03**		**7**	**2.43E−03**	**7.71E−04**		**0**
CLPSO	6.96E+03	1.49E+03		0	1.86E−02	7.92E−03		0
PSO-w	4.88E+02	3.51E+02		0	1.64E−02	6.28E−03		0
wFIPS-PSO	1.47E+03	5.40E+02		0	8.61E−03	2.12E−03		0
FDR-PSO	3.35E+02	2.37E+02		0	9.74E−03	4.48E−03		0
PSO-cf-local	5.96E+02	3.86E+02		0	6.84E−03	2.05E−03		0
PSO-cf	2.39E+02	2.97E+02		0	6.07E−03	2.83E−03		0
PSO-CL-pbest	6.88E+03	1.18E+03		0	1.33E−01	1.04E−01		0
CPSO-H	2.54E+04	1.15E+04		0	5.30E−02	2.17E−02		0

performs worse than PSO-cf-local on functions 1, 3 and 4, because SLPSO needs some generations to find the most suitable strategy. It is observed in Fig. 2 that the initial convergence speed of SLPSO on these functions is not so high, but after a learning period, the convergence speed of SLPSO increases and becomes comparable with PSO-cf. This trend is much clearer for functions 3 and 5, on which SLPSO provides the best performance. Especially, it should be noted that, for function 5 (Rosenbrock function), a well-known hard test problem, only SLPSO approaches the true global optimum through the extremely narrow valley of the fitness landscape, while the other algorithms are still struck after the final generation.

The functions 6–11 are separate multi-modal problems, where the number of local minima increases exponentially as the dimension of the function increases. Functions 8 and 9 (original and non-continuous Rastrigin problems) are the hardest ones with the deepest local optima among these problems. It is shown in Table 6 that only SLPSO, CLPSO and CPSO-H succeed in converging to the global optimum. In general, CLPSO slightly outperforms SLPSO on these two problems. This is because the updating strategy of CLPSO is the most suitable for the complex multi-modal problems, but SLPSO spends some computational cost to perform selection on candidate updating strategies. However, once the basin containing the global optimum is found, SLPSO abruptly accelerates, see Figs. 2 and 3. For the other problems, SLPSO always provides the best performance except for function 7. That means the cooperation of different strategies in SLPSO works quite well. The same conclusion can be obtained from Figs. 2 and 3 where the convergence speed of only SLPSO on these problems increases as the optimization process goes on, that is, SLPSO is able to adaptively tune the probabilities of using different strategies to suit different problems and different optimization stages.

5.4.2. Experimental results on groups 2 and 3

The test functions of groups 2 and 3 are extended forms of the three hard functions of group 1. As discussed in the above subsection, the uni-modal problem by Rosenbrock and the multi-modal problem by Rastrigin are two typical hard tasks and we generalize three mis-scaled test problems of these two functions from group 2, which is similar to [18]. The mis-scaling between the longest and shortest axis is 10 for Rastrigin10 and 100 for Rastrigin100 and Rosenbrock100. For the 30 dimensional problems, the factor between "adjacent" axes is 1.08 and 1.17, respectively. In group 3, the only function is the generalized form of function 2 (Schwefel 1.2). The systematical Gaussian noise $0.4|N(0,1)|$ makes it difficult to accurately locate the global optimum.

For group 2, the performance of all algorithms deteriorates, compared with the corresponding basic test functions of group 1. It is observed in Table 7 that the mean values of all algorithms are beyond 10^2. Still, SLPSO is also the only algorithm that succeeds in finding the global optimum in 17 runs. According to the results for the Rastrigin functions, Rastrigin10 and

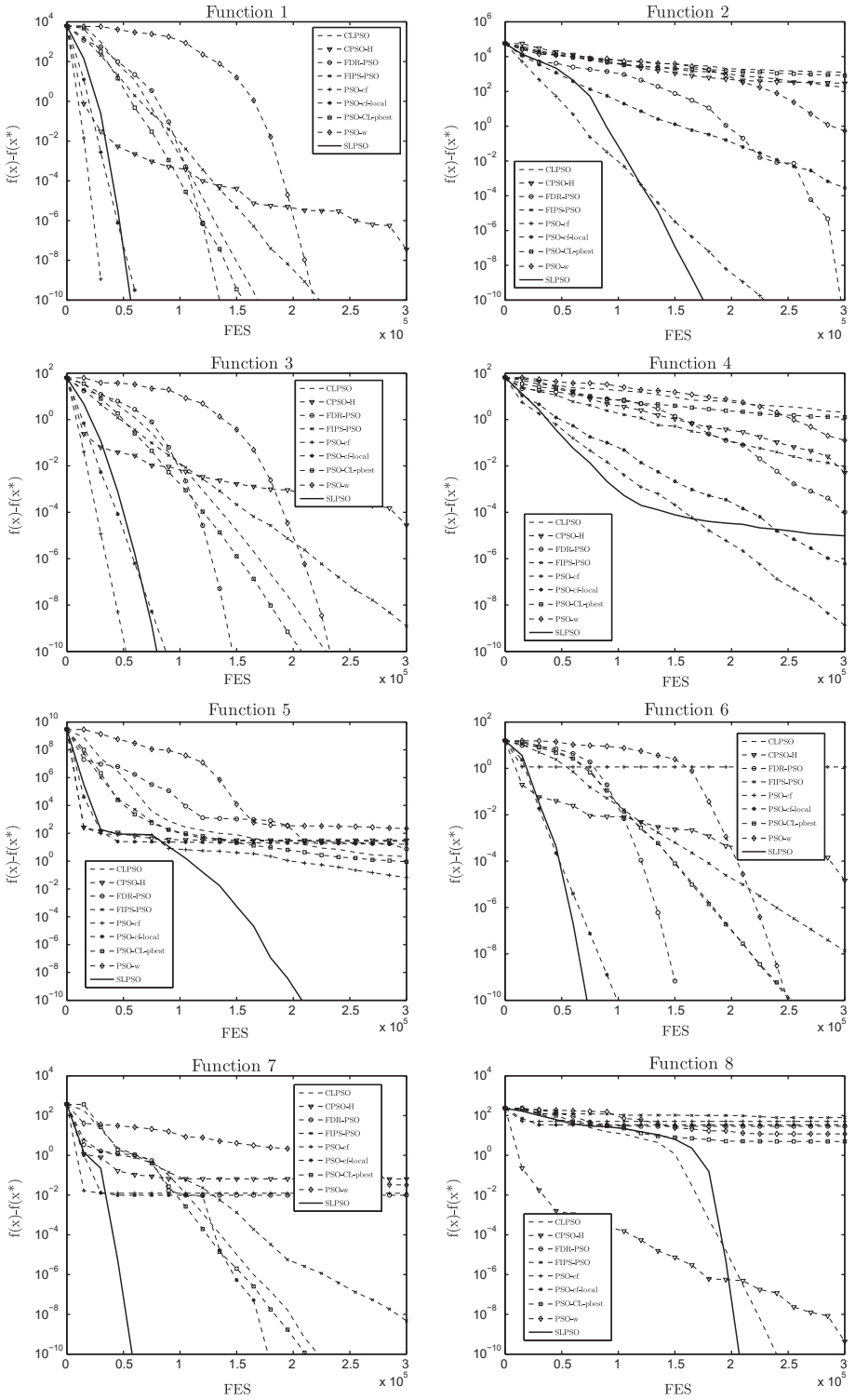

Fig. 2. Optimization curves of function optimization with 30 *D*. (Sphere, Schwefel 1.2, Schwefel 2.22, Schwefel 2.21, Rosenbrock, Achkey, Griewank and Rastrigin).

Rastrigin100, the performances of SLPSO and CLPSO are steady while CPSO-H tends to fail as the mis-scaling increases. It can be concluded that the one-dimensional decompose-and-conquer strategy of CPSO-H cannot effectively assign the optimization strength for the variables with different contributions to the fitness value. The convergence characteristics of these problems are similar to the non-mis-scaled problems, see Fig. 3.

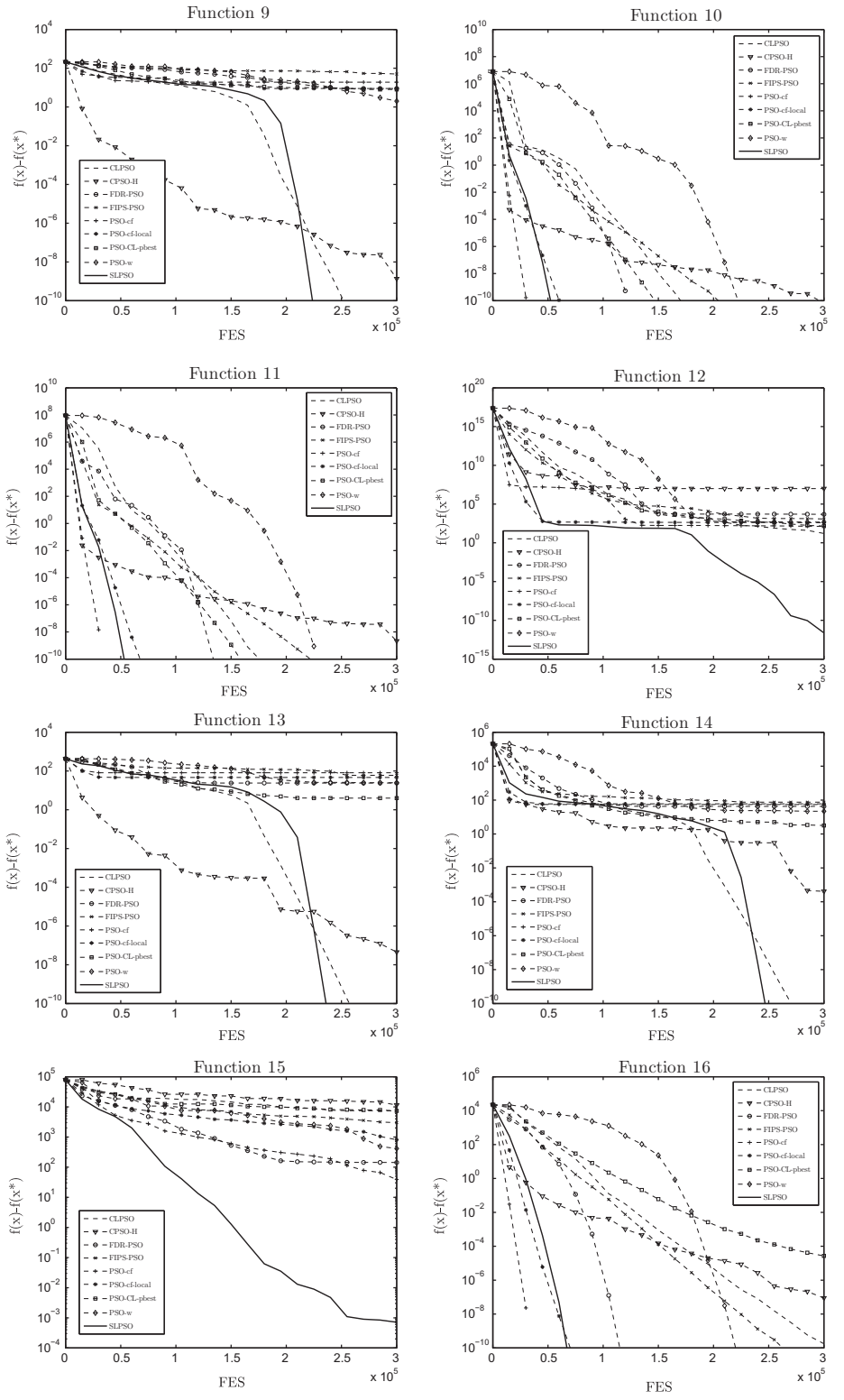

Fig. 3. Optimization curves of function optimization with 30 *D*. (Noncontinuous Rastrigin, Penalized 1, Penalized 2, Rosenbrock 100, Rastrigin 10, Rastrigin 100, Noise Schewefel 1.2, Rotated Sphere).

The noisy function 14 demands intensive exploitation of the algorithms. Some algorithms with good exploration ability, such as CLPSO, wFIPS-PSO, PSO-CL-pbest and CPSO-H, perform badly on this problem. Although SLPSO's performance is also

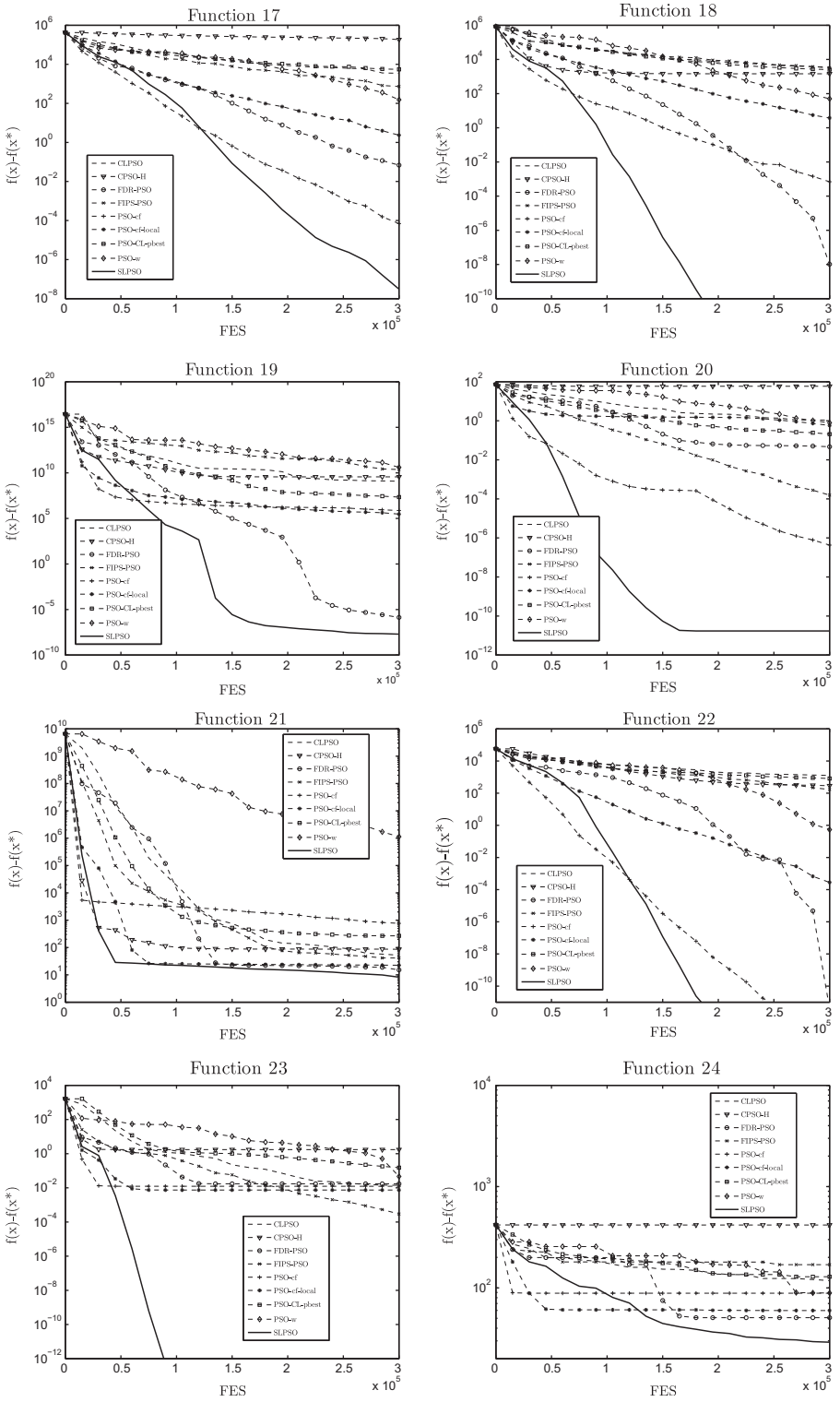

Fig. 4. Optimization curves of function optimization with 30 *D*. (Rotated Tablet, Rotated Ellipse, Rotated diff pow, Rotated Schwefel 2.21, Rotated Rosenbrock, Rotated Ackley, Rotated Griewank, Rotated Rastrigin).

affected by the noisy landscape, it outperforms the other algorithms by at least four magnitudes, see Table 7. It can be observed in Fig. 3 that the SLPSO provides much higher convergence speed than the other algorithms, whose convergence characteristics are identical.

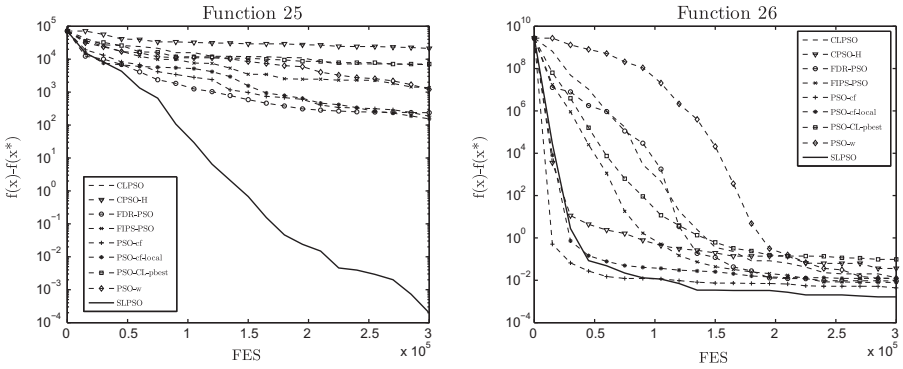

Fig. 5. Optimization curves of function optimization with 30 *D*. (Rotated Noise Schwefel 1.2, Rotated Noise Quadric).

5.4.3. Experimental results on group 4

The experiment on group 4 is designed to test the capability of handling non-separability and ill-condition. Functions 17–19 are convex-quadratic and can be transformed into the Sphere problem. These functions are mis-scaled, with the axises scaling between the longest and shortest principal axis are 1000, 20 and 10. Besides, two more typical uni-modal problems, Schwefel 2.21 and Rosenbrock are also used. Functions 23–25 are multi-modal problems that are extended from the original Ackley, Griewank and Rastrigin functions. The last two functions are rotated noisy functions with systematical or white noise.

From the results of functions 17–26 in Tables 7 and 8, it is interesting to see that SLPSO provides the best results for almost all problems with function 17, the simplest problem, as only exception. It is not surprising that only SLPSO uses the DbV, which is a specific strategy for rotated problems. It is worth being noted that SLPSO succeeds in finding the global optimum of function 25, a rotated noisy Schwefel problem, in 7 runs, while all of the other algorithms completely fail. Besides, even for the extremely hard tasks, such as functions 21, 24 and 26, SLPSO also outperforms the other algorithms in terms of solution quality, see Table 8. The reason for this behavior is likely that SLPSO is able to choose the most suitable strategy for different search stages. That is, for the hard tasks that always require good cooperation of different strategies, the learning mechanism of SLPSO makes it possible to use EbV and DbV to shrink the search space when the search region is smooth, and to use CLPSO and PSO-CL-pbest to handle the complex fitness landscapes. Fig. 5 clearly illustrates that the convergence speed of SLPSO is remarkably higher than that of the other algorithms on uni-modal, multi-modal and even noisy problems. According the results obtained by CPSO-H, its performance is seriously affected by the non-separable characteristic and only the rotated Sphere problem can be solved by it.

5.4.4. Overall performance comparison

In this subsection, we benchmark the performance of all algorithms in two aspects: (a) the empirical cumulative distribution of normalized *SP* and (b) *t*-test. The empirical distribution of normalized *SP* is generated as follows: (1) choose the results of all functions where at least one algorithm was successful in at least one run, in other words, the functions 15, 24 and 26 are excluded; (2) the *SP*s of all nine algorithms on each test function are calculated; (3) all *SP*s are divided by the best *SP* on respective function. The figures of empirical cumulative distribution for uni-modal problems only, multi-modal problems only and all problems are plotted in Fig. 6. In these figures, small values of normalized *SP* and large values of the empirical distribution are preferable. The first one that reaches the top of the graph will be considered as the best algorithm [43]. Moreover, the results of a one-tailed *t*-test with 98° of freedom at a 0.05 level of significance are provided to statistically show the comparison between SLPSO and the other algorithms. In Table 9, the *t*-test results regarding *algorithm*1 vs. *algorithm*2 are shown as '+', '−', '=', 's+' and 's−' when *algorithm*1 is insignificantly better than, insignificantly worse than, equal to, significantly better than, and significantly worse than *algorithm*2 respectively.

It is observed from Fig. 6 that only the overall performance curve of SLPSO on uni-modal problems reaches the top (100%), which means all considered uni-modal problems can be solved by SLPSO. For the multi-modal problems, the situation is similar, but when the empirical cumulative distribution curve of SLPSO reaches the top, the normalized *SP* value is still very small. That means SLPSO provides top convergence speed on almost all multi-modal functions. In summary of the above two aspects, the overall performance of SLPSO on all test functions is also the best, see Fig. 6. The excellent performance of SLPSO is additionally validated in Table 9. Especially on the hard problems of group 4, SLPSO significantly outperform the other algorithms.

The experimental results show that SLPSO provides the best performance for both uni-modal and multi-modal problems, which require conflicting search capabilities, exploration and exploitation. Does this fact mean that no free lunch theory [15] does not hold in this experiment? The answer is No. In the framework of SLPSO, it is observed that four updating strategies aiming at handling problems with different characteristics and even different optimization stages of one problems are used in parallel. The learning mechanism integrates these strategies as a whole, and extracts the most suitable strategies from the

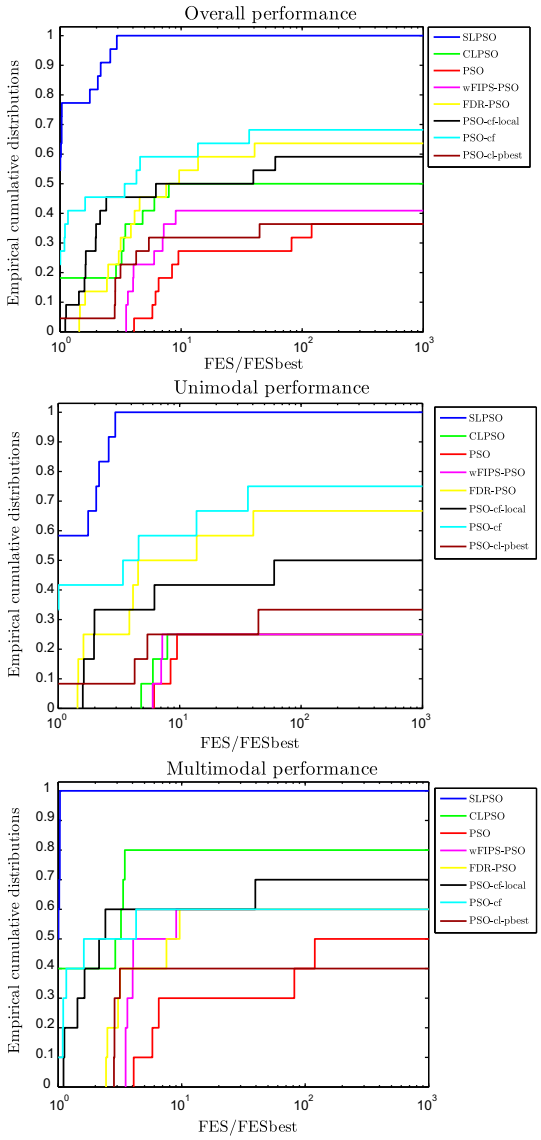

Fig. 6. Empirical distribution of normalized success performance on all test functions, unimodal functions only and multi-modal functions only.

experiences. In this case, the successful rate of SLPSO highly depends on the robustness of the learning mechanism. For example, the convergence characteristics on function 4 illustrate that although SLPSO converges fastest at the beginning, the learning mechanism assigns large probability to the wrong strategy, which results in a degeneration of the convergence speed as the search continues, see Fig. 2. Nonetheless, the learning mechanism is powerful enough for most problems. Therefore, the no free lunch theory still holds, and the essence of SLPSO is to strengthen the universality and robustness by using the learning mechanism.

5.5. Analysis of the self-adaptive learning mechanism

In order to further investigate the effectiveness of self-adaptive learning mechanism of SLPSO, the execution probabilities of all four separate updating strategies on four typical problems are plotted in Fig. 7. The probability curves are the mean of 30 independent runs, whose statistical results have been recorded in the above subsection. The most suitable strategy for a given problem at a given point in time during the search should yield the largest probability.

As presented in Section 3, the purpose of the EbV strategy is to quickly decrease the search region when the fitness landscape is relatively simple. It is observed in Fig. 7 that for the separable uni-modal problem Sphere, EbV occupies the most proportion before the global optimum is found (before $FES = 5 \times 10^4$). After that, the probability of using DbV increases shar-

Table 9
The *t*-test results of comparing SLPSO with the other algorithms.

t-test	Group1								
	fun1	fun2	fun3	fun4	fun5	fun6	fun7	fun8	fun9
SLPSO vs CLPSO	=	s+	=	s+	s+	s+	s−	=	=
SLPSO vs PSO-w	=	s+	=	s+	s+	s+	s+	s+	s+
SLPSO vs wFIPS-PSO	=	s+	s+	s+	s+	s+	s−	s+	s+
SLPSO vs FDR-PSO	=	s+	=	s+	s+	s+	s+	s+	s+
SLPSO vs PSO-cf-local	=	s+	=	−	s+	=	s+	s+	s+
SLPSO vs PSO-cf	=	s+	s+	s−	s+	=	s+	s+	s+
SLPSO vs PSO-CL-pbest	=	s+	=	s+	s+	s+	s−	s+	s+
SLPSO vs CPSO-H	s+	s+	s+	s+	s+	s+	s+	s+	s+
	Group1		Group2 and 3				Group4		
	fun10	fun11	fun12	fun13	fun14	fun15	fun16	fun17	fun18
SLPSO vs CLPSO	=	=	s+	=	=	s+	=	s+	s+
SLPSO vs PSO-w	s+	=	s+	s+	s+	s+	s+	s+	s+
SLPSO vs wFIPS-PSO	=	s+	s+	s+	s+	s+	=	s+	s+
SLPSO vs FDR-PSO	=	=	s+	s+	s+	s+	s+	s+	s+
SLPSO vs PSO-cf-local	s+	s+	s+	s+	s+	s+	=	s+	s+
SLPSO vs PSO-cf	s+	s+	s+	s+	s+	s+	=	s+	s+
SLPSO vs PSO-CL-pbest	=	=	s+	s+	s+	s+	=	s+	s+
SLPSO vs CPSO-H	s+	s+	s+	s+	s+	s+	s+	s+	s+
	Group4								
	fun19	fun20	fun21	fun22	fun23	fun24	fun25	fun26	
SLPSO vs CLPSO	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs PSO-w	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs wFIPS-PSO	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs FDR-PSO	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs PSO-cf-local	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs PSO-cf	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs PSO-CL-pbest	s+	s+	s+	s+	s+	s+	s+	s+	
SLPSO vs CPSO-H	s+	s+	s+	s+	s+	s+	s+	s+	

ply, because when all particles converge to the global optimum, DbV always generates the new offsprings locating in the global optimum due to the loss of the difference information. It is interesting to see that the curves of probabilities on Rosenbrock are similar to those on Sphere. After a learning period, about a quarter of the optimization process, DbV occupies the most proportion. However, Fig. 2 illustrates that the optimization process of SLPSO on Rosenbrock function lasts $FES = 1.4 \times 10^5$, which is remarkably longer than that on Sphere ($FES = 5 \times 10^4$). That means DbV plays a crucial role in optimizing Rosenbrock, which verifies that DbV has superior performance on ill-conditioned problems. The same behavior is also observed on the rotated diff pow problem, which also has linkage relationship among the variables. However, the strategy EbV, which is suitable for the non-rotated uni-modal problems, receives small probability just after a few generations. The reason is that EbV adopts a local search strategy, which is so effective on the rotated problems. The Rastrigin problem is a typical multi-modal problem, whose 2-D fitness landscape is shown in Fig. 1. On this problem, the probabilities of using EbV and DbV quickly lower down to near 0. Since CLPSO and PSO-CL-pbest yield better performance, the self-adaptive learning mechanism assigns more probabilities to them. When the basin that contains the global optimum is found ($FES > 2 \times 10^5$), the problem degenerates to be a uni-modal problem. In this case, the execution probabilities of EbV and DbV tends to increase, because the self-adaptive learning mechanism can adaptively tune the probabilities based on the experience at different optimization stages.

6. Experimental study on ELD optimization

Economic load dispatch (ELD) optimization is an important and difficult task in designing superior power systems [57]. As introduced in literature [11,57], the goal of optimizing ELD problems is to search an optimal operating condition for the generator units of a power plant in order to minimize their operational costs while satisfy constraints. In other words, the ELD optimization approaches have to satisfy the power demands of all customers with the pursuit of minimum fuel cost. In this experiment, we focus on large scale problems with both, valve points and multiple fuels, for which scarce papers have been published [11,35,62].

6.1. Problem definition

Without loss of generality, ELD optimization can be stated as a minimization problem as follows [57]:

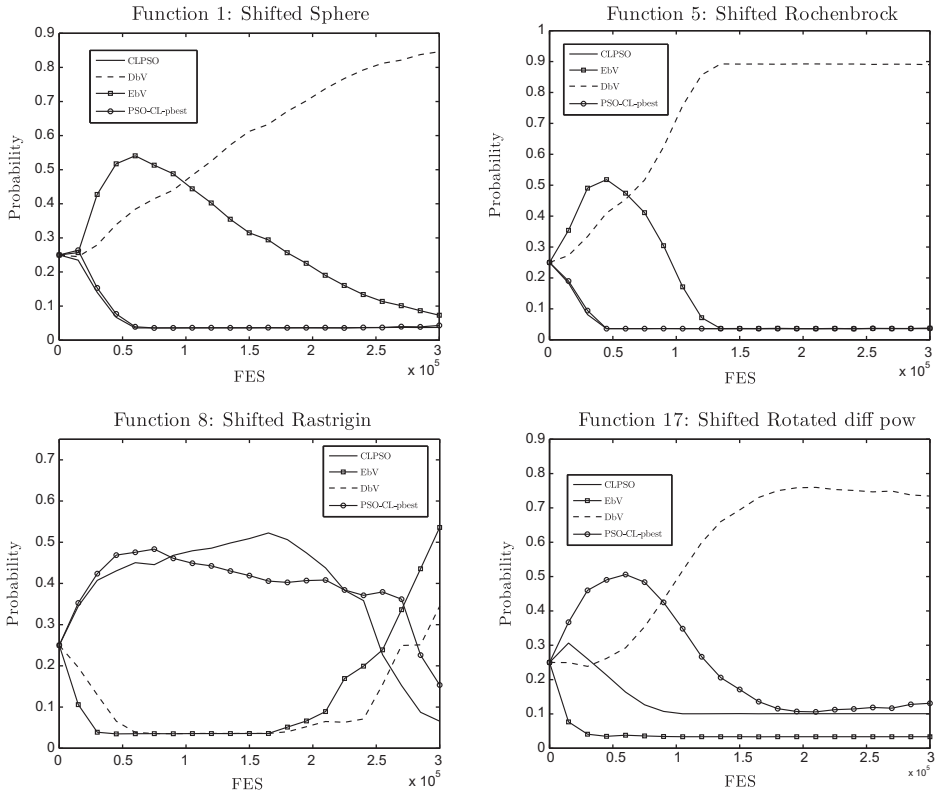

Fig. 7. Self-adaptive learning characteristics of the probabilities of using different strategies. (Sphere, Rosenbrock, Rastrigin, Rotated diff pow).

$$\begin{aligned} &\text{minimize } C = \sum_{j\in J} F_j(P_j) \\ &\text{subject to } D = \sum_{j\in J} P_j, \\ &\qquad P_{j\min} \leqslant P_j \leqslant P_{j\max} \quad \text{for } j = 1, 2, \ldots, n. \end{aligned} \tag{23}$$

The variables in Eq. (23) are defined as follows:

C	total generator cost
J	set for all generators
F_j	cost function of jth generator
D	total demand of J
$P_{j\min}$	the minimum output of generator j
$P_{j\max}$	the maximum output of generator j
P_j	electrical output of generator
n	number of generators

In formal terms, ELD problem can be defined as minimizing the total cost C subject to power balance constraints $D = \sum_{j\in J} P_j$ and generating capacity constraints $P_{j\min} \leqslant P_j \leqslant P_{j\max}$. The most simplified cost function of each generator can be represented as a quadratic function as expressed in Eq. (24) [11,62], whose solution can be obtained by conventional mathematical optimization methods:

$$F_j(P_j) = a_j + b_j P_j + c_j P_j^2, \tag{24}$$

where a_j, b_j, c_j are the cost coefficients of generator j.

To create a more precise and practical ELD simulation, the valve point effect and the effects of multiple fuels should also be considered into the formulation of ELD problems [11]. The definition of ELD problems with both valve points and multiple fuels can be represented as follows [11,35]:

$$F_j(P_j) = \begin{cases} a_{j1} + b_{j1}P_j + c_{j1}P_j^2 + \left|e_{j1} \cdot \sin(f_{j1}) \cdot \left(P_{j1}^{\min} - P_{j1}\right)\right|, & \text{for fuel } l\ P_j^{\min} \leqslant P_j \leqslant P_{j1} \\ a_{j2} + b_{j2}P_j + c_{j2}P_j^2 + \left|e_{j2} \cdot \sin(f_{j2}) \cdot \left(P_{j2}^{\min} - P_{j2}\right)\right|, & \text{for fuel } 2\ P_{j1} \leqslant P_j \leqslant P_{j2} \\ \ldots \\ a_{jk} + b_{jk}P_j + c_{jk}P_j^2 + \left|e_{jk} \cdot \sin(f_{jk}) \cdot \left(P_{jk}^{\min} - P_{jk}\right)\right|, & \text{for fuel } k\ P_{jk-1} \leqslant P_j \leqslant P_j^{\max}, \end{cases} \tag{25}$$

where P_{ji} is the ith breakpoint for the jth generator. With this formulation, the simulation can be much more practical compared with the above two kinds of simulation.

6.2. Experimental results and discussion

The test instances used in our experiments are directly cited from [11]. On the 10-generator problem, we compare SLPSO with DE [35], PSO [35], the classical GA with multiple update strategy (CGA-MU) [11], and the improved GA with multiple update strategy (IGA-MU) [11]. In existing research, few methods have been proposed for problems involving more than 40 generators. The reason may be that the large scale ELD problems with such complex conditions are too hard to solve [62]. Two important methods in this area are IGA-MU and CGA-MU. To make the comparison fair to these two algorithms, the computational cost used by SLPSO is guaranteed to be not higher than any of the compared algorithms. With this guarantee, the results of the compared algorithms are directly cited from the above associated references. The statistical results of 50 runs on the 10-generator problem are summarized in Table 10. Table 11 depicts the frequency of attaining a cost within the specific goal ranges out of 100 runs for each algorithm with 100 different initial trial solutions. The statistical results of 100 runs on 20, 40, 80 and 160-generator problems are summarized in Table 12.

Table 10 reveals that SLPSO is superior to the other algorithms in terms of all comparison aspects. This is especially true when looking into Table 11 that SLPSO is able to obtain final solutions with significantly better quality than the other algorithms. The steady characteristic is not satisfactory for the other algorithms, because no run of other algorithms can obtain a solution within the region [623.5–624.5].

Similar to the 10-generator case, SLPSO shows better performance than that of GA-based algorithms on problems with more generators. It is clearly shown in Table 12 that SLPSO provides top performance for all numbers of generators setting. Especially, more than $ 100 are saved in the 40 and 160-generator problems, compared with the solutions obtained with the GA-based algorithms. It is strongly indicated that SLPSO is definitely very suitable for complex large scale ELD optimization.

Table 10
Statistical comparison on the 10-generator problem with both valve points and multiple fuels. The unit of cost is $.

Algorithm	SLPSO	DE	RGA	PSO	CGA-MU	IGA-MU
Minimum cost	**623.94**	624.51	624.51	624.51	627.61	624.52
Maximum cost	**624.00**	624.56	624.51	624.51	633.87	630.87
Mean cost	**623.84**	624.52	624.51	624.51	624.72	625.87

Table 11
Relative Frequency of convergence in percent for the 10-generator problem with both valve points and multiple fuels. The unit of cost is $.

Algorithm	623.5–624.5	624.5–625.5	625.5–626.5	626.5–627.5	627.5–628.5	628.5–629.5	629.5–630.5	630.5–631.5	631.5–632.5	632.5–633.5	633.5–634.5
SLPSO	100	0	0	0	0	0	0	0	0	0	0
DE	0	100	0	0	0	0	0	0	0	0	0
RGA	0	100	0	0	0	0	0	0	0	0	0
PSO	0	100	0	0	0	0	0	0	0	0	0
CGA-MU	0	5	20	31	21	10	7	3	2	0	1
IGA-MU	0	39	45	11	2	2	0	1	0	0	0

Table 12
Scalability comparison on problem with both valve points and multiple fuels. The number of generators scales from 10 to 160. The unit of cost is $.

Algorithm	10	20	40	80	160
No. of units (mean cost)					
SLPSO	**623.94**	**1247.80**	**2377.64**	**4992.71**	**10012.68**
CGA-MU	627.61	1249.39	2500.92	5008.14	10143.73
IGA-MU	625.87	1249.12	2499.82	5003.88	10042.47

7. Conclusion

In order to strengthen the universality and robustness of PSO, this paper proposes a self-adaptive learning based particle swarm optimization (SLPSO). In SLPSO, four updating strategies aiming at handling problems with different characteristics at different stages of the optimization process of one problem are combined in parallel in a self-adaptive learning mechanism, which is used to tune the execution probabilities based on the experience of the past generations.

The advantages of SLPSO are experimentally verified via comparing it with eight state-of-the-art PSO variants on 26 test functions with different characteristics such as uni-modality, multi-modality, rotation, ill-condition, mis-scale and noise. The reasons leading to good performance of SLPSO have been discussed in depth. Moreover, a practical engineering problem, the economic load dispatch problem for power systems, is also used to assess the performance of SLPSO on real-word engineering problems. In this ELD experiment, SLPSO updates the best solution records for large scale problems with both valve points and multiple fuels.

This work is also expected to call for more attention on using self-adaptive learning based mechanisms in swarm intelligence. There are still several problems remaining to be investigated, such as how to more appropriately make use of the information gathered during past generations of an optimization process and how to apply the algorithms of this kind to more real-world engineering problems.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants No. 60401015, U0835002), the Chinese Academy of Science (CAS) Special Grant for Postgraduate Research, Innovation and Practice, IEEE Walter Kaplus Student Summer Research Grant, and Graduate Innovation Fund of University of Science and Technology of China.

The authors thank Mr. Suganthan for providing the source codes of his group. The authors also would like to thank Xuexiao Lai, Xiaolei Chen and Ruoxi Xiang for their invaluable contributions.

References

[1] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC), Anchorage, 1998, pp. 84–89.
[2] A. Arcuri, X. Yao, Search based software testing of object-oriented containers, Information Sciences 178 (15) (2008) 3075–3095.
[3] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, Proceeding in IEEE Congress on Evolutionary Computation (CEC) (2005) 1769–1776.
[4] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[5] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.
[6] T.M. Blackwell, P.J. Bentley, Dont push me! collision-avoiding swarms, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC), Honolulu, HI, 2002, pp. 1691–1696.
[7] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, R. Qu, A survey of hyper-heuristics, Tech. Rep. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham, 2009.
[8] A. Cervantes, I.M. Galvn, P. Isasi, AMPSO: a new particle swarm method for nearest neighborhood classification, IEEE Transactions on Systems, Man, and Cybernetics 39 (5) (2009) 1082–1091.
[9] F.-C. Chang, H.-C. Huang, A refactoring method for cache-efficient swarm intelligence algorithms, Information Sciences (2010), doi:10.1016/j.ins.2010.02.025.
[10] Chen DeBao, Zhao ChunXia, Particle swarm optimization with adaptive population size and its application, Applied Soft Computing 9 (2009) 39–48.
[11] C. Chiang, Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels, IEEE Transactions on Power Systems 20 (4) (2005) 1690–1699.
[12] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[13] P. Cowling, K. Chakhlevitch, Hyperheuristics for managing a large collection of low level heuristics to schedule personnel, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC2003), Canberra, Australia, 2003, pp. 1214–1221.
[14] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach for scheduling a sales summit, in: Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000, Lecture Notes in Computer Science, Springer, Konstanz, Germany, 2000, pp. 176–190.
[15] J.C. Culberson, On the futility of blind search an algorithmic view of no free lunch, Evolutionary Computation 6 (2) (1998) 109–127.
[16] W.L. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Sciences 178 (15) (2008) 3096–3109.
[17] C.K. Goh, K.C. Tan, D.S. Liu, S.C. Chiam, A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design, European Journal of Operational Research 202 (1) (2010) 42–54.
[18] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[19] Sheng-Ta Hsieh, Tsung-Ying Sun, Chan-Cheng Liu, Shang-Jeng Tsai, Efficient population utilization strategy for particle swarm optimizer, IEEE Transactions on Systems, Man, and Cybernetics 39 (2) (2009) 444–456.
[20] Chia-Feng Juang, Chi-Yen Wang, A self-generating fuzzy system with ant and particle swarm cooperative optimization, Expert Systems with Applications 36 (2009) 5362–5370.
[21] A. Kaveh, S. Talatahari, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, Computers and Structures 87 (2009) 267–283.
[22] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, Proceeding of the Congress on Evoluationary Computation (1999) 1931–1938.
[23] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceeding of the IEEE International Conference on Neural Networks 4 (1995) 1942–1948.
[24] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufman, 2001.

[25] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceeding of the IEEE Congress on Evoluationary Computation, Honolulu, HI, 2002, pp. 1671–1676.
[26] P. Larranga, A review on estimation of distribution algorithms, in: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer, 2001, pp. 57–100 (Chapter 3).
[27] P. Larranga, R. Etxeberria, J.A. Lozano, J.M. Pena, Optimization in continuous domains by learning and simulation of Gaussian networks, in: Proceeding in 2000 Genetic and Evolutionary Computation Conference Workshop Program, 2000, pp. 201–204.
[28] 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 (3) (2006) 281–295.
[29] J.J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceeding in IEEE Swarm Intell. Symposium Pasadena, CA, 2005, pp. 68–75.
[30] H.W. Liu, J. Li, A particle swarm optimization-based multiuser detection for receive-diversity-aided STBC systems, IEEE Signal Processing Letters 15 (2008) 29–32.
[31] M. Lovbjerg, T. Krink, Extending particle swarm optimizers with self-organized criticality, in: Proceeding of the Congress on Evoluationary Computation, Honolulu, HI, 2002, pp. 1588–1593.
[32] Qiang Luo, Dongyun Yi, A co-evolving framework for robust particle swarm optimization, Applied Mathematics and Computation 199 (2008) 611–622.
[33] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Information Sciences 180 (2010) 1571–1581.
[34] R. Mallipeddi, P.N. Suganthan, Differential evolution algorithm with ensemble of populations for global numerical optimization, Opsearch 46 (2) (2009) 184–213.
[35] P.S. Manoharan, P.S. Kannan, S. Baskar, M.W. Iruthayarajan, Penalty parameter-less constraint handling scheme based evolutionary algorithm solutions to economic dispatch, IET Generation, Transmission and Distribution 2 (4) (2008) 478–490.
[36] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 204–210.
[37] V. Miranda, N. Fonseca, New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control, in: Proceeding of the 14th Power Syst. Comput. Conf., Seville, Spain, 2002 [online]. Available: <http://www.pscc02.org/papers/s21pos.pdf>.
[38] S. Nema, J. Goulermas, G. Sparrow, P. Cook, A hybrid particle swarm branch-and-bound (HPB) optimizer for mixed discrete nonlinear programming, IEEE Transactions on Systems, Man, and Cybernetics 38 (6) (2008) 1411–1427.
[39] K.E. Parsopoulos, M.N. Vrahatis, UPSOA unified particle swarm optimization scheme, Proceeding in Lecture Series on Computational Sciences (2004) 868–873.
[40] K.E. Parsopoulos, M.N. Vrahatis, Parameter selection and adaptation in unified particle swarm optimization, Mathematical and Computer Modelling 46 (2007) 198–213.
[41] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, Proceeding in Swarm Intelligence Symposium (2003) 174–181.
[42] A.M. Potter, K.A.D. Jong, A cooperative coevolutionary approach to function optimization, in: Proceeding in the Third International Conference on Parallel Problem Solving from Nature, Springer-Verlag, 1994, pp. 249–257.
[43] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[44] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, Proceeding in IEEE Congress on Evolutionary Computation (CEC) (2005) 1785–1791.
[45] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time varying accelerating coefficients, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 240–255.
[46] V. Robles, Jose M. Peña, Pedro Larrañaga, María S. Pírez, GA-EDA: a new hybrid cooperative search evolutionary algorithm, in: J.A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation, Studies in Computational Intelligence, vol. 192, Springer-Verlag, 2006, pp. 187–219.
[47] M. Sebag, A. Ducoulombier, Extending population-based incremental learning to continuous search spaces, Proceeding in Parallel Problem Solving from Nature (PPSN VI) (1998) 418–427.
[48] P.S. Shelokar, Patrick Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Applied Mathematics and Computation 188 (2007) 129–142.
[49] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC), 1998, pp. 69–73.
[50] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceeding of the 7th Conference Evoluationary Programming, New York, 1998, pp. 591–600.
[51] Y. Shi, R.C. Eberhart, Particle swarm optimization with fuzzy adaptive inertia weight, in: Proceeding of the Workshop Particle Swarm Optimization, Indianapolis, 2001, pp. 101–106.
[52] P.N. Suganthan, Particle swarm optimizer with neighborhood operator, in: Proceeding of the Congress Evoluationary Computation, Washington, DC, 1999, pp. 1958–1962.
[53] 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 for CEC2005 special session, 2005.
[54] J. Sun, Q.F. Zhang, E. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Information Sciences 169 (2005) 249–262.
[55] K. Tang, X. Yao, Special Issue on "Nature inspired problem-solving", Information Sciences 178 (15) (2008) 2983–2984.
[56] Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, Sankar Kumar Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Information Sciences 177 (2007) 5033–5049.
[57] V.D. Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.C. Hernandez, R.G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems, IEEE Transactions on Evolutionary Computation 12 (2) (2008) 171–195.
[58] Jasper A. Vrugt, Bruce A. Robinson, Improved evolutionary optimization from genetically adaptive multimethod search, Proceedings of the National Academy of Sciences USA 104 (3) (2007) 708–711.
[59] Jasper A. Vrugt, Bruce A. Robinson, James M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 243–259.
[60] S. Wang, J. Watada, A hybrid modified PSO approach to VaR-based facility location problems with variable capacity in fuzzy random uncertainty, Information Sciences (2010), doi:10.1016/j.ins.2010.02.014.
[61] Y. Wang, B. Li, A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization, in: Raymond Chiong (Ed.), Nature-Inspired Algorithms for Optimization, Studies in Computational Intelligence, vol. 193, Springer-Verlag, 2009, pp. 171–198.
[62] Y. Wang, B. Li, Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems, Information Sciences 180 (2010) 2405–2420.
[63] Z.Y. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178 (15) (2008) 2985–2999.
[64] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

[65] Peng-Yeng Yin, Fred Glover, Manuel Laguna, Jia-Xian Zhu, Cyber swarm algorithms-improving particle swarm optimization using adaptive memory strategies, European Journal of Operational Research 201 (2) (2010) 377–389.
[66] B. Yuan, Marcus Gallagher, Experimental results for the special session on real-parameter optimization at CEC 2005: a simple continuous EDA, Proceeding in IEEE Conference on Evolutionary Computation (2005) 1792–1799.
[67] M. Zhang, W.J. Luo, X.F. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178 (15) (2008) 3043–3074.
[68] Zhao Xinchao, A perturbed particle swarm algorithm for numerical optimization, Applied Soft Computing 10 (1) (2010) 119–124.
[69] Yuxin Zhao, Wei Zub, Haitao Zeng, A modified particle swarm optimization via particle visual modeling analysis, Computers and Mathematics with Applications 57 (2009) 2022–2029.