A Self-Learning Particle Swarm Optimizer for Global Optimization Problems

Changhe Li, Shengxiang Yang *Member, IEEE*, and Trung Thanh Nguyen

***Abstract*—Particle swarm optimization (PSO) has been shown as an effective tool for solving global optimization problems. So far, most PSO algorithms use a single learning pattern for all particles, which means all particles in a swarm use the same strategy. This monotonic learning pattern may cause the lack of intelligence for a particular particle, which makes it unable to deal with different complex situations. This paper presents a novel algorithm, called self-learning particle swarm optimizer (SLPSO), for global optimization problems. In SLPSO, each particle has a set of four strategies to cope with different situations in the search space. The cooperation of the four strategies is implemented by an adaptive learning framework at the individual level, which can enable a particle to choose the optimal strategy according to its own local fitness landscape. The experimental study on a set of 45 test functions and two real-world problems show that SLPSO has a superior performance in comparison with several other peer algorithms.**

***Index Terms*—Self-learning particle swarm optimization (SLPSO), particle swarm optimization (PSO), operator adaptation, topology adaptation, global optimization problem.**

I. Introduction

EVOLUTIONARY computation has become an important active research area over the past several decades. In the literature, global optimization benchmark problems have become more and more complex, from simple unimodal functions to rotated shifted multi-modal functions to hybrid composition benchmark functions proposed recently [41]. Finding the global optima of a function has become much more challenging or even practically impossible for many problems. Hence, far more effective optimization algorithms are always needed.

In the past decade, PSO has been actively studied and applied for many academic and real-world problems with promising results [33]. However, many experiments have shown that the basic PSO algorithm easily falls into local optima when solving complex multi-modal problems [25] with a huge number of local optima. In the literature of PSO, for most algorithms so far, all particles in a swarm use only a single learning pattern. This may cause the lack of intelligence for a particle to cope with different complex situations. For example, different problems may have different properties due to different shapes of the fitness landscapes. In order to effectively solve these problems, particles may need different learning strategies to deal with different situations. This may also be true even for a specific problem because the shape of a local fitness landscape in different sub-regions of a specific problem may be quite different, such as the composition benchmarks in [41].

To bring more intelligence to each particle, an adaptive learning PSO algorithm (ALPSO) that utilizes four strategies was introduced in [21], where each particle has four learning sources to serve the purpose of exploration or exploitation during the search process. An adaptive technique proposed in [22] was used to adjust a particle's search behavior between exploration and exploitation by choosing the optimal strategy.

Based on our previous work in [21], a novel algorithm, called self-learning PSO (SLPSO), is proposed in this paper. Compared with ALPSO, some new features are introduced in SLPSO. Firstly, two strategies in ALPSO are replaced by two new strategies. Secondly, we use a biased selection method in SLPSO so that a particle learns from only the particles whose $pbest$s are better than its $pbest$. Thirdly, the method of updating the frequency parameter in ALPSO is replaced by a new one. Fourthly, we introduce a super particle generated by extracting promising information from improved particles instead of directly updating the $gbest$ particle, where the extracting method is also enhanced by using a learning ratio. Fifthly, controlling the number of particles that learn from the super particle is implemented. Finally, although some new parameters are introduced in SLPSO, users do not need to tune these parameters for a specific problem as they use the same setting up methods for all problems.

The rest of this paper is organized as follows. Section II describes the basic PSO algorithm and some variants. The SLPSO algorithm is presented in Section III. Section IV describes the test problems and experimental setup. The experimental study on SLPSO, including the self-learning mechanism and sensitivity analysis of parameters, is presented in Section V . The performance comparison of SLPSO with some peer algorithms taken from the literature is given in Section VI. Finally, Section VII concludes this paper.

II. Related Work

A. Particle Swarm Optimization

Similar to other evolutionary algorithms (EAs), PSO is a population based stochastic optimization technique. A potential solution in the fitness landscape is called a particle in PSO.

Manuscript received June 01, 2011; accepted September 23, 2011. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U. K. under Grant EP/E060722/1 and Grant EP/E060722/2.

C. Li is with the School of Computer Science, China University of Geosciences, Wuhan, 430074, China (email: changhe.lw@gmail.com).

S. Yang is with the Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, U. K. (email: shengxiang.yang@brunel.ac.uk).

T. T. Nguyen is with The School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, Liverpool L3 3AF, U. K. (email: T.T.Nguyen@ljmu.ac.uk).

Each particle i is represented by a position vector $\vec{x}_i$ and a velocity vector $\vec{v}_i$, which are updated as follows:

$$v'^d_i = \omega v^d_i + \eta_1 r_1(x^d_{pbest_i} - x^d_i) + \eta_2 r_2(x^d_{gbest} - x^d_i) \quad (1)$$

$$x'^d_i = x^d_i + v'^d_i, \quad (2)$$

where x'^d_i and x^d_i represent the current and previous positions in the d-th dimension of particle i, respectively, v'_i and v_i are the current and previous velocity of particle i, respectively, $\vec{x}_{pbest_i}$ and $\vec{x}_{gbest}$ are the best position found by particle i so far and the best position found by the whole swarm so far, respectively, $\omega \in (0, 1)$ is an inertia weight, which determines how much the previous velocity is preserved; η_1 and η_2 are the acceleration constants, and r_1 and r_2 are random numbers generated in the interval $[0.0, 1.0]$ uniformly. Without loss of generality, minimization problems are considered in this paper.

There are two main models of the PSO algorithm, called $gbest$ (global best) and $lbest$ (local best), respectively. The two models differ in the way of defining the neighborhood for each particle. In the $gbest$ model, the neighborhood of a particle consists of the particles in the whole swarm, which share information between each other. On the contrary, in the $lbest$ model, the neighborhood of a particle is defined by several fixed particles. The two models give different performances on different problems. Kennedy and Eberhart [19] and Poli *et al.* [33] pointed out that the $gbest$ model has a faster convergence speed but also has a higher probability of getting stuck in local optima than the $lbest$ model. On the contrary, the $lbest$ model is less vulnerable to the attraction of local optima, but has a slower convergence speed than the $gbest$ model. In order to give a standard form for PSO, Bratton and Kennedy proposed a standard version of PSO (SPSO)[1] in [5]. In SPSO, a local ring population topology is used and the experimental results have shown that the $lbest$ model is more reliable than the $gbest$ model on many test problems.

B. Some Variant Particle Swarm Optimizers

Due to its simplicity and effectiveness, PSO has become a popular optimizer and many improved versions have been reported in the literature since it was first introduced. Most studies address the performance improvement of PSO from one of four aspects: population topology [25], [18], [20], [24], [29], [40], diversity maintenance [3], [4], [6], [21], [34], hybridization with auxiliary operations [1], [32], [46], and adaptive PSO [15], [34], [36], [37], [52], which are briefly reviewed below.

1) Population Topology: The population topology has a significant effect on the performance of PSO. It determines the way particles communicate or share information with each other. Population topologies can be divided into static and dynamic topologies. For static topologies, communication structures of circles, wheels, stars, and randomly-assigned edges were tested in [18], showing that the performance of algorithms is different on different problems depending on the topology used. Then, Kennedy and Mendes [20] have tested a large number of aspects of the social-network topology on five test functions. After that, a fully informed PSO (FIPS) algorithm was introduced by Mendes [29]. In FIPS, a particle uses a stochastic average of $pbest$s from all of its neighbors instead of using its own $pbest$ position and the $gbest$ position in the update equation. A recent study in [24] showed that PSO algorithms with a ring topology are able to locate multiple global or local optima if a large enough population size is used.

For dynamic topologies, Suganthan [40] suggested a dynamically adjusted neighbour model, where the search begins with a $lbest$ model and is gradually increased until the $gbest$ model is reached. Janson *et al.* proposed a dynamic hierarchical PSO (H-PSO) [16] to define the neighborhood structure where particles move up or down the hierarchy depending on the quality of their $pbest$ solutions. Liang and Suganthan [25] developed a comprehensive learning PSO (CLPSO) for multimodal problems. In CLPSO, a particle uses different particles' historical best information to update its velocity, and for each dimension a particle can potentially learn from a different exemplar.

2) PSO with Diversity Control: Ratnaweera *et al.* [34] stated that the lack of population diversity in PSO algorithms is a factor that makes them prematurely converge to local optima. Several approaches of diversity control have been introduced in order to avoid the whole swarm converging to a single optimum. In [3], diversity control was implemented by preventing too many particles from getting crowded in one region of the search space. Negative entropy was added into PSO in [49] to discourage premature convergence. Other researchers have applied multi-swarm methods to maintain diversity in PSO. In [6], a niching PSO (NichePSO) was proposed by incorporating a cognitive only PSO model with the guaranteed convergence PSO (GCPSO) algorithm [45]. Parrott and Li developed a speciation based PSO [23], which dynamically adjusts the number and size of swarms by constructing an ordered list of particles, ranked according to their fitness, with spatially close particles joining a particular species. An atomic swarm approach was adapted to track multiple optima simultaneously with multiple swarms in dynamic environments by Blackwell and Branke [4]. Recently, a clustering PSO algorithm was proposed in [50], where a hierarchical clustering method is used to produce multi-swarms in promising regions in the search space.

3) Hybrid PSO: Hybrid EAs are becoming more and more popular due to their capabilities in handling problems that involve complexity, noisy environments, imprecision, uncertainty, and vagueness. Among hybrid EAs, hybrid PSO is an attractive topic. The first hybrid PSO algorithm was developed by Angeline [1], where a selection scheme is introduced. In [46], the fast evolutionary programming (FEP) [51] was modified by replacing the Cauchy mutation with a version of PSO velocity. Hybrid PSO based on genetic programming was proposed in [32]. A cooperative PSO (CPSO-H) algorithm was proposed in [44], which employs the idea of splitting the search space into smaller solution vectors and combines with the $gbest$ model of PSO. In [7], a hybrid model that integrates

[1]Notice that the term "SPSO" in this paper indicates the standard version of PSO in [5] rather than the speciation-based PSO in [23], which will be introduced later.

PSO, recombination operator, and dynamic linkage discovery, called PSO-RDL, was proposed.

4) PSO with Adaptation: Besides the above three active research aspects, adaptation is another promising research trend in PSO. Shi and Eberhart introduced a method of linearly deceasing ω with the iteration for PSO in [36], and then proposed a fuzzy adaptive ω method for PSO in [37]. Ratnaweera *et al.* [34] developed a self-organizing hierarchical PSO with time-varying acceleration coefficients, where η_1 and η_2 are set to a large value and a small value, respectively, at the beginning, and are gradually reversed during the search process. By considering a time-varying population topology, FIPS's velocity update mechanism [29], and a decreasing inertia weight, a PSO algorithm, called Frankenstein's PSO (FPSO), was proposed in [11]. In FPSO, the swarm starts with a fully connected population topology. The connectivity decreases with a certain pattern and ends up with the ring topology. The performance of a TRIBES model of PSO was investigated in [8], which is able to automatically change the behavior of particles as well as the population topology. A population manager method for PSO was proposed in [15], where the population manager can dynamically adjust the number of particles according to some heuristic conditions defined in [15].

Recently, a PSO version with adaptive ω, η_1, and η_2, called APSO, was proposed by Zhan *et al.* [52]. In APSO, four evolutionary states, including "exploitation", "exploration", "convergence", and "jumping out", are defined. Co-incidentally, the four operators in ALPSO [21] play the similar roles as the four evolutionary states defined in APSO [52], but the way of implementation is different. While the four operators in ALPSO are updated by an operator adaptation scheme [21], the evolutionary states in APSO are estimated by evaluating the population distribution and particle fitness. In each evolutionary state, APSO gives one corresponding equation to adjust the value of η_1 and η_2. Accordingly, the value of ω is tuned using a sigmoid mapping of the evolutionary factor f in each evolutionary state. Similar work of adaptively tuning the parameters of PSO has been done in [48], [31]. However, none or little work of operator selection has been done in PSO. Experimental study in [21] has shown that the adaptive learning approach is an effective method to improve the performance of PSO.

C. Self-learning and Adaptive Strategies

Swarm intelligence (SI) is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge [47]. Agents (individuals) have some simple interaction rules and each agent is able to adjust its behavior according to the results of interaction with local environments. Complex, emergent, and collective behaviors are shown up only based on simple interactions with local environments. However, it is difficult for us to understand and simulate the emergent behaviors from nature to solve real-world problems.

Although it is difficult to exactly simulate the emergent collective behaviors, we can employ techniques based on adaptive operator selection (AOS) to implement the adaptive behavior of social creature groups, e.g., ant colony, bee colony, and bird flocking, etc. AOS is an important step toward self-tuning for EAs. To design an efficient AOS structure, generally, we need to solve two issues: a credit assignment mechanism, which computes a reward for each operator at hand based on certain rules from statistical information of offspring; and an adaptation mechanism, which is able to adaptively select one among different operators based on their rewards [9], [12], [14], [39], [38], [42], [43].

The low-level behavior of the choice-function based hyper-heuristic was investigated in [17]. Based on the fact that no single operator is optimal for all problems and the optimal choice of operators for a given problem is also time-variant, Smith and Fogarty [39] suggested a framework for the classification based on the learning strategy used to control them and reviewed a number of adaptation methods in GAs. Thereafter, to address the issue that the set of available operators may change over time, Smith [38] proposed a method for estimating an operator's current utility, which is able to avoid some of the problems of noise inherent in simpler schemes for memetic algorithms. In [42], [43], an adaptive allocation strategy, called the adaptive pursuit method, was proposed and compared with some other probability matching approaches in a controlled, dynamic environment. In [9], a specific dynamic method based on the multi-armed bandit paradigm was developed for dynamic frameworks. In order to well evaluate the performance of operators, recently, a new strategy [27] was introduced by considering not only the fitness improvements from parent to offspring, but also the way they modify the diversity of the population, and their execution time.

III. Self-Learning Particle Swarm Optimizer

A. General Considerations of Performance Tradeoff

It is generally believed that the $gbest$ model biases more toward exploitation, while the $lbest$ model focuses more on exploration. Although there are many improved versions of PSO, the question of how to balance the performance of the $gbest$ and $lbest$ models is still an important issue, especially for multi-modal problems.

In order to achieve good performance, a PSO algorithm needs to balance its search between the $lbest$ and $gbest$ models. However, this is not an easy task. If we let each particle simultaneously learn from both its $pbest$ position and the $gbest$ or $lbest$ position to update itself (velocity update), the algorithm may suffer from the disadvantages of both models. One solution might be to implement the cognition component and the social component separately. This way, each particle can focus on exploitation by learning from its individual $pbest$ position or focus on convergence by learning from the $gbest$ particle. The idea enables particles of different locations in the fitness landscape to carry out local search or global search, or vice versa, in different evolutionary stages. So, different particles can play different roles (exploitation or convergence) during the search progress, and even the same particle can play different roles during different search progress.

Algorithm 1 Update(operator i, particle k, fes)

if $i = a$ **then**
  Update the velocity and position of particle k using operator a and Eq. (2);
else if $i = b$ **then**
  Update the position of particle k using operator b;
else if $i = c$ **then**
  Choose a random particle j that is not particle k;
  if $f(\vec{x}_{pbest_j}) < f(\vec{x}_{pbest_k})$ **then**
    Update the velocity and position of particle k using operator c and Eq. (2);
  else
    Update the velocity and position of particle j using operator c and Eq. (2);
    $k := j$;
  end if
else
  Update the velocity and position of particle k using operator d and Eq. (2);
end if
fes++; where fes is the current number of fitness evaluations.

However, there is a difficulty in implementing this idea: the appropriate moment for a particle to learn from $gbest$ or $pbest$ is very hard to know. To implement the above idea, we introduce an adaptive method to automatically choose one search strategy. In this method, which strategy to apply and when to apply that strategy are determined by the property of the local fitness landscape where a particle is located. This adaptive method will be further described in the following sections.

B. Velocity Update in SLPSO

Inspired by the idea of division of labor, we can assign different roles to particles, e.g., converging to the global best particle, exploiting the personal best position, exploring new promising areas, or jumping out of local optima. Accordingly, we summarize four different possible situations regarding surrounding environments for a general particle. Firstly, for unimodal problems or the fitness landscape with peaks that are far away from each other, to effectively search on local peaks, the best learning policy for all particles may be to learn from their neighbour best particles. Secondly, for a particle in a slope, the best strategy may be to learn from its own $pbest$ position as it will help the particle to find a better position much easier than searching in a distant sub-region. Thirdly, for the fitness landscape with many local optima evenly distributed, learning from neighbors may be the best choice as this strategy will encourage particles to explore the search space. Finally, the only choice for converged particles is to apply mutation to jump out of local optima. Of course, mutation can also help particle to explore the search space.

Based on the above analysis, we define four strategies and four corresponding operators in SLPSO. In SLPSO, the learning information for each particle comes from four sources: the archived position of the $gbest$ particle ($abest$), which is the same as the "super" particle introduced in Section II, its individual $pbest$ position, the $pbest$ position of a random particle ($pbest_{rand}$) whose $pbest$ is better than its own $pbest$, and a random position $prand$ nearby. The four strategies play the roles of convergence, exploitation, exploration, and jumping out of the basins of attraction of local optima, respectively.

The four strategies enable each particle to independently deal with different situations. For each particle k, the learning equations corresponding to the four operators, respectively, are given as follows:

- Operator a: learning from its $pbest$ position

$$exploitation: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (pbest_k^d - x_k^d) \quad (3)$$

- Operator b: learning from a random position nearby

$$jumping\ out: x_k^d = x_k^d + v_{avg}^d \cdot N(0,1) \quad (4)$$

- Operator c: learning from the $pbest$ of a random particle

$$exploration: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (pbest_{rand}^d - x_k^d) \quad (5)$$

- Operator d: learning from the $abest$ position

$$convergence: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (abest^d - x_k^d) \quad (6)$$

where $pbest_{rand}$ is the $pbest$ of a random particle, which is better than $pbest_k$; the jumping step v_{avg}^d is the average speed of all particles in the d-th dimension, which is calculated by $v_{avg}^d = \sum_{k=1}^{N} |v_k^d|/N$, where N is the population size; $N(0,1)$ is a random number generated from the normal distribution with mean 0 and variance 1; the $abest$ position is an archive of the best position found by SLPSO so far.

It should be noted that different from ALPSO [21], a bias selection scheme is added into the operator of learning from the $pbest_{rand}$ position in SLPSO. A particle only learns from a $pbest_{rand}$ position that is better than its own historical best position $pbest$. Due to this scheme, more resources are given to the badly-performing particles to improve the whole swarm. The procedure is described in Algorithm 1.

It should also be noted that the $abest$ position in Eq. (6) is an archive of the position of the $gbest$ particle, which is different from the $gbest$ particle of the whole swarm because it is not a particle. The $abest$ position does not use any of the four operators to update itself except Algorithm 2 (to be explained later in Section III-D). Although it is the same position as the $gbest$ particle in the initial population, it will be updated by Algorithm 2 and becomes better than the $gbest$ particle. Different from the ALPSO algorithm in [21], all particles in SLPSO, including the $gbest$ particle, learn from the $abest$ position. The position and velocity update framework in SLPSO is shown in Algorithm 1.

The third operator, which is a new one used in this paper, enables a particle to explore the non-searched areas with a higher probability than learning from its nearest neighbor as used in [21]. From Eq. (5), learning from different particles can actually alter the velocity as the distance is different from different random particles in two successive iterations. Hence, this strategy is able to maintain the social diversity of the swarm at a certain level.

The choice of which learning option is the most suitable would depend on the local fitness landscape where a particle is located. However, it is assumed that we cannot know how the fitness landscape looks like even though we have *a priori* knowledge of the fitness landscape. Instead, each particle should detect the shape of the local fitness landscape where it is currently in by itself. How to achieve this goal is described in the following section.

C. The Adaptive Learning Mechanism

As analyzed above, the best strategy for a particular particle is determined by its local fitness landscape. Therefore, the optimal strategy for a particle may change in accordance with the change of its position during the evolution process. In this section, we will achieve two objectives. The first is to provide a solution to how to choose the optimal strategy, and the other one is to adapt this learning mechanism to the local environmental change for a particular particle during the evolutionary process.

From another point of view, the four operators in SLPSO actually represent four different population topologies, which in turn allow particles to have four different communication structures. Each population structure determines a particle's neighborhood and the way it communicates with the neighborhood. By adaptively adjusting the population topology, we can adaptively adjust the way particles interact with each other and hence can enable the PSO algorithm to perform better in different situations.

The task of selecting operators from a set of alternatives has been comprehensively studied in EAs [12], [14], [39]. Inspired by the idea of probability matching [42], in this paper, we introduce an adaptive framework using the aforementioned four operators, each of which is assigned to a selection ratio. This adaptive framework is an extension of our work in [22]. Different from the work in [22], where the adaptive scheme is only implemented at the population level, in this paper, we extend the adaptive scheme to the individual level and make it simpler than the version in [21], [22]. The adaptive framework is based on the assumption that the most successful operator used in recent past iterations may also be successful in the future several iterations. The selection ratio of each operator is equally initialized to $1/4$ for each particle and is updated according to its relative performance.

For each particle, one of the four operators is selected according to their selection ratios. The operator that results in a higher relative performance, which is evaluated by a combination of the offspring fitness, current success ratio, and previous selection ratio, will have its selection ratio increased. Gradually, the most suitable operator will be chosen automatically and control the learning behavior of each particle in different evolutionary stages and local fitness landscapes. During the updating period for each particle, the progress value and the reward value of operator i are calculated as follows.

The progress value $p_i^k(t)$ of operator i for particle k at iteration t is defined as:

$$p_i^k(t) = \begin{cases} \mid f(\vec{x}_k(t)) - f(\vec{x}_k(t-1)) \mid, & \text{if operator } i \text{ is chosen} \\ & \text{by } \vec{x}_k(t) \text{ and } \vec{x}_k(t) \text{ is better than } \vec{x}_k(t-1) \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

The reward value $r_i^k(t)$ has three components, which are the normalized progress value, the success rate, and the previous selection ratio. It is defined as:

$$r_i^k(t) = \frac{p_i^k(t)}{\sum_{j=1}^{R} p_j^k(t)}\alpha + \frac{g_i^k}{G_i^k}(1-\alpha) + c_i^k s_i^k(t) \tag{8}$$

where g_i^k is the counter that records the number of successful learning times of particle k, in which its child is fitter than particle k by applying operator i since the last selection ratio update, G_i^k is the total number of iterations where operator i is selected by particle k since the last selection ratio update, g_i^k/G_i^k is the success ratio of operator i for particle k, α is a random weight between 0.0 and 1.0, R is the number of operators, c_i^k is a penalty factor for operator i of particle k, which is defined as follows:

$$c_i^k = \begin{cases} 0.9, & \text{if } g_i^k = 0 \text{ and } s_i^k(t) = \max_{j=1}^{R}(s_j^k(t)) \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

and $s_i^k(t)$ is the selection ratio of operator i for particle k at the current iteration.

In Eq. (8), if none of the operators has been able to improve particle k since the last selection ratio update, then $\sum_{j=1}^{R} p_j^k(t)$ will be 0. In this case, only the third component ($c_i^k s_i^k(t)$) will be assigned to $r_i^k(t)$.

Based on the above definitions, the selection ratio of operator i for particle k in the next iteration $t+1$ is updated as follows:

$$s_i^k(t+1) = \frac{r_i^k(t)}{\sum_{j=1}^{R} r_j^k(t)}(1 - R * \gamma) + \gamma, \tag{10}$$

where γ is the minimum selection ratio for each operator, which is set to 0.01 for all the experiments in this paper.

According to the above definitions, we know that there is always one operator that has the highest selection ratio for each particle. This operator must be the most successful one compared with the other operators at the current moment. However, when a particle converges or moves to a new local sub-region whose property is different from the previous one, this most successful operator no longer brings any benefit to the particle. When this case occurs, according to the punishing mechanism in Eq. (9), the selection ratio of that operator will decrease, while the selection ratios of the other operators will increase. So, a new most suitable operator will be adaptively selected based on its relatively better performance and the outdated operator will lose its domination automatically. This is how the adaptive mechanism works. Based on the analysis of the adaptive working mechanism, we can see that SLPSO is able to choose the optimal strategy and is also able to adapt with the environmental change for each particle independently.

In addition, it should be noted that the selection ratios of the four operators are updated at the same time and not updated every iteration. Instead of counting the number of successive iterations for U_f, called update frequency in [21], we just record the number of successive unsuccessful learning times (m_k) for each particle k. If particle k is improved by any operator before the counter m_k reaches the maximal value of U_f, m_k will be reset to 0. This method reduces the risk of punishing the best operator due to its temporally bad performance in a short period. After the selection ratios of the four learning operators are updated, all the information, except the selection ratio of each operator, is reset to 0.

D. Information Update for the $abest$ Position

In most population-based algorithms, once an individual is updated or replaced, the information of all dimensions will be replaced with that of a new position. This updating

Algorithm 2 UpdateAbest(particle k,fes)

```
for each dimension d of abest do
    if rand() < P_l^k then
        x_{t_abest} := x_abest;
        x_{t_abest}[d] := x_k[d];
        Evaluate x_{t_abest};
        fes++;
        if f(x_{t_abest}) < f(x_abest) then
            x_abest[d] := x_{t_abest}[d];
        end if
    end if
end for
```

Algorithm 3 UpdateLearningOpt(particle k)

```
if CF_k != true && PF_k = true then
    sum := sum_{j=1}^{3} s_j^k;
    for j := 1 to 3 do
        s_j^k := s_j^k / sum;
    end for
    s_4^k := 0;
end if
if CF_k = true && PF_k != true then
    for j := 1 to 4 do
        p_j^k := 0; g_j^k := 0; G_j^k := 0; s_j^k := 1/4
    end for
end if
```

where CF_k and PF_k are used to record whether particle k uses the convergence operator or not at the current and previous iteration, respectively

mechanism has one disadvantage: promising information may not always be preserved. For example, even though an individual has promising information in a particular dimension, that information would still be lost if it has a lower fitness due to unpromising information from other dimensions. This problem, called "two step forward, one step back" in [44], has two opposite aspects to be considered. One is that the improvement of some dimensions at the gene level brings the improvement of the whole individual at the individual level. The other aspect is that some dimensions get worse although the whole individual gets better.

To overcome the above two issues, we should monitor the improved particles in PSO. If a particle gets better over time, there may be the case that the particle has some useful information in some certain dimensions even though it has a relatively low fitness value. In that case, other particles should learn from that useful information. In SLPSO, the $abest$ position learns the useful information from the dimensions of particles which show improvement over time.

However, it is difficult to effectively implement this idea. For example, in order to check the information of which dimension of an improved particle is useful for the $abest$ position, we have to check all the dimensions of that improved particle. This is because it is very difficult to know the information of which dimension or combination of dimensions of the improved particle is useful for the $abest$ position. Although we do not know that information, we can assign a learning probability (P_l) for each dimension to the $abest$ position to learn from the improved particle. There are two advantages to introduce the learning probability: firstly, the algorithm will save a lot of computational resources, and secondly, the algorithm can reduce the probability of learning potentially useless information for the $abest$ position even if it also reduces the probability of learning useful information. Actually, for most cases, the information of an improved particle may be not useful for the $abest$ position when they distribute in different sub-regions. Therefore, assigning a learning probability may not affect too much the performance of SLPSO. On the contrary, if the $abest$ position learns potentially useless or even dangerous information from improved particles, it will seriously affect SLPSO's performance. The evidence can be seen from the experimental results later in the section of parameter sensitivity analysis. Algorithm 2 describes the update framework of the $abest$ position.

E. Controlling the Number of Particles That Learn from the $abest$ Position

Learning from the $abest$ position is used to accelerate the population convergence. For the particles close to the $abest$ position, the attraction to the $abest$ position is too strong to give any opportunity to the other three operators. In order to make full use of the adaptive learning mechanism, we need to control the number of particles that learn from the $abest$ position. There are two reasons to do so.

First, the optimal number of particles that learn from the $abest$ position is determined by the property of the problem to be solved. For example, to effectively solve the unimodal function, e.g., the Sphere function, we need to allow all particles to learn from the $abest$ position so that all particles can quickly converge to the global optimum. On the contrary, for some multi-modal problems, we should allow most particles to do local search rather than to learn from the $abest$ position so that they will have a higher probability to find the global optimum. The evidence can be seen from the experimental results in the section of parameter sensitivity analysis.

Second, it is not fair for the other three operators to compete with the convergence operator because all of them contribute to the convergence operator. When a particle gets improvement through whichever of the other three operators, the convergence operator also gets benefit through either a direct or an indirect way: in the direct way, the improved particle becomes the new $abest$ position, and in the indirect way, useful information is extracted from improved particles and hence, if the $abest$ position succeeds, it will also indirectly get benefit.

Based on the above analysis, we need to control the number of particles to use the convergence operator. However, it is hard to know which particles are suitable to use the convergence operator. To solve this problem, we randomly select a certain number of particles to use the convergence operator every iteration. To implement this idea, we need to update some information of the particles that switch between using and not using the convergence operator in two successive iterations. The information that needs to be updated includes progress values, reward values, success ratios, and selection ratios.

If the switch happens, there are two cases for updating the related information. The first case is that particles use the convergence operator in the previous iteration but do not use it in the current iteration, and the second case is opposite to the first one. In the first case, we need to remove the learning source of

Algorithm 4 The SLPSO Algorithm

```
Generate initial swarm and set up parameters for each particle;
Set fes :=0, iteration counter for initial swarm t := 0;
while fes < T_Fes do
    for each particle k do
        Select one learning operator i using the roulette wheel selection rule;
        Update(i, k, fes);
        G_i^k ++;
        if f(x_k(t)) < f(x_k(t-1)) then
            g_i^k ++; m_k := 0;
            p_i^k += f(x_k(t-1)) - f(x_k(t));
            Perform UpdateAbest(k, fes) for the abest position;
        else
            m_k := m_k + 1;
        end if
        if f(x_k(t)) < f(x_pbest_k) then
            x_pbest_k := x_k;
            if f(x_k) < f(x_abest) then
                x_abest := x_k;
            end if
        end if
        if m_k ≥ U_f^k then
            Update the selection ratios according to Eq. (10);
            for each operator j do
                p_j^k := 0; g_j^k := 0; G_j^k := 0;
            end for
        end if
    end for
    UpdatePar();
    t++;
end while
```

the $abest$ position, and then re-normalize the selection ratios of the other three operators according to their current values and keep other information of the three operators the same. For the second case, all the related information is reset to the initial states: the progress values, reward values, and success ratios are set to 0, and the selection ratios are set to 1/4 for the four operators. The update description can been seen in Algorithm 3.

F. Framework of SLPSO

Together with the above components, the implementation of the SLPSO algorithm is summarized in Algorithm 4. The procedure of $UpdatePar()$ in Algorithm 4 will be introduced later in Algorithm 5 in Section V-C.

1) $Vmax$ and Out of Search Range Handling in SLPSO: In SLPSO, we use the parameter $Vmax$ to constrain the maximum velocity for each particle, and the value of $Vmax$ is set to the half of the search domain for a general problem. We use the following operation to handle out-of-range search: for each dimension, before updating its position, we first remember the position value $x^d(t-1)$ of the previous iteration, then calculate a temporal value x_t by Algorithm 1, and finally, update its current position $x^d(t)$ as follows:

$$x^d(t) = \begin{cases} R(X^d_{min}, x^d(t-1)), & \text{if } x_t < X^d_{min} \\ R(x^d(t-1), X^d_{max}), & \text{if } x_t > X^d_{max} \\ x_t, & \text{else} \end{cases} \tag{11}$$

where $R(a, b)$ returns a uniformly distributed number within the range $[a, b]$ and $[X^d_{min}, X^d_{max}]$ is the search range in the d-th dimension of a given problem.

2) Complexity of SLPSO: Compared with the basic PSO algorithm, SLPSO needs to perform some extra computation on updating the selection ratios, the $abest$ position, and the three parameters. For the update of selection ratios and parameters, we do not need to re-evaluate the fitness values of particles, and the time complexity is $O(N)$ (where N is the population size) for each iteration. For the $abest$ position update, although it needs to re-evaluate particle's fitness, the re-evaluation happens only when particles get improvement. In addition, for each dimension of the $abest$ position, the update is performed with a certain probability. According to the above component complexity analysis, we can see that the time complexity of SLPSO is not very high in comparison with the basic PSO algorithm.

IV. Test Functions and Experimental Setup

A. Test Functions

To investigate how SLPSO performs in different environments, we chose 45 functions, including the traditional functions, traditional functions with noise, shifted functions, and rotated shifted functions, which are widely used in the literature [29], [25], [51], [52] as well as the complex hybrid composition functions proposed recently in [26], [41]. The details of these functions are given in Table I, Table II, and Table III, respectively.

The 45 functions are divided into six groups in terms of their properties: traditional problems (f_1-f_{12}), noisy problems (f_{19}-f_{22}), shifted problems (f_{15}-f_{18}, f_{31}-f_{32}, and f_{36}-f_{38}), rotated problems (f_{23}-f_{26}), rotated shifted problems (f_{27}-f_{30}, f_{33}-f_{35}, and f_{39}), and hybrid composition functions (f_{13}-f_{14} and f_{40}-f_{45}). Table IV shows the parameter settings for some particular functions. The shifting and rotating methods used in the test functions are from [41]. The detailed parameter setting for functions f_{31}-f_{45} can be found in [41].

In order to generate noisy environments, the functions f_{19}-f_{22} are modified from four traditional test functions by adding noises in each dimension as follows:

$$f(\vec{x}) = g(\vec{x} - 0.01 \cdot \vec{o}_r) \tag{12}$$

where $\vec{o}_r$ is a vector of uniformly distributed random numbers within the range $[0, 1]$.

It should be noted that the test functions chosen from the literature are not in favor of the tested algorithms, including SLPSO. For example, beside the solved cases from [41] in the competition of CEC 2005, we also chose some never-solved problems, e.g., the composition functions f_{40}-f_{45}. Moreover, they have different properties to test an algorithm's performance in terms of different aspects, such as unimodal problems (e.g., f_1, f_8, and f_9-f_{11}), problems of a huge number of local optima (e.g., f_2), non-continuous problems (e.g., f_3), non-differentiable problems (e.g., f_4), non-separate problems (e.g., f_5), deceptive problems (e.g., f_6), functions with noise (e.g., f_{19}), problems with modified fitness landscape (e.g.,f_{15}, f_{25}, and f_{28}), and problems with very complex fitness landscapes (e.g., f_{13}).

B. Parameter Settings for the Involved PSO Algorithms

The configuration of each peer algorithm taken from the literature is given in Table V, which is exactly the same as that used in the original paper. Below, we briefly describe the

TABLE I
THE TEST FUNCTIONS, WHERE f_{min} IS THE MINIMUM VALUE OF A FUNCTION AND $S \in R_n$

Name	Test Function	S	f_{min}
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
Rastrigin	$f_2(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12, 5.12]	0
Noncont_Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$	[-5.12, 5.12]	0
Weierstrass	$f_4(\vec{x}) = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20$	[-0.5,0.5]	0
Griewank	$f_5(\vec{x}) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	[-600, 600]	0
Schwefel	$f_6(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500, 500]	0
Ackley	$f_7(\vec{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	[-32, 32]	0
Rosenbrock	$f_8(\vec{x}) = \sum_{i=1}^{n} 100(x_{i+1}^2 - x_i)^2 + (x_i - 1)^2)$	[-2.048, 2.048]	0
Schwefel_2_22	$f_9(\vec{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	0
Schwefel_1_2	$f_{10}(\vec{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100, 100]	0
Schwefel_2_21	$f_{11}(\vec{x}) = \max_{i=1}^{n} \lvert x_i \rvert$	[-100, 100]	0
Penalized_1	$f_{12}(\vec{x}) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4), y_i = 1 + (x_i + 1)/4$	[-50, 50]	0
H_Com	$f_{13}(\vec{x})$ =Hybrid Composition function (CF4) in [26]	[-5, 5]	0
RH_Com	$f_{14}(\vec{x})$ =Hybrid Composition function (CF4) with rotation in [26]	[-5, 5]	0

TABLE II
TEST FUNCTIONS OF f_{15} TO f_{30}, WHERE "O" REPRESENTS THE ORIGINAL PROBLEMS, "N", "S", "R", AND "RS" REPRESENT THE MODIFIED PROBLEMS BY ADDING NOISE, SHIFTING, ROTATING, AND COMBINATION OF SHIFTING AND ROTATING, RESPECTIVELY

	O	N	S	R	RS		O	N	S	R	RS
Sphere	f_1	f_{19}	f_{18}	f_{23}	f_{27}	Schwefel	f_6	f_{20}	f_{15}	f_{25}	f_{28}
Rastrigin	f_2	f_{22}	f_{17}	f_{24}	f_{30}	Ackley	f_7	f_{21}	f_{16}	f_{26}	f_{29}

TABLE III
TEST FUNCTIONS OF f_{31} TO f_{45} CHOSEN FROM [41]

f	Name	S	f_{min}
f_{31}	S_Sphere_CEC05(F_1)	[-100,100]	-450
f_{32}	S_Rastrigin_CEC05(F_9)	[-5,5]	-330
f_{33}	RS_Rastrigin_CEC05(F_{10})	[-5,5]	-330
f_{34}	RS_Weierstrass_CEC05(F_{11})	[-0.5,0.5]	90
f_{35}	RS_Ackley_Bound_CEC05(F_8)	[-32,32]	-140
f_{36}	S_Rosenbrock_CEC05(F_6)	[-100,100]	390
f_{37}	S_Schwefel_1_2_CEC05(F_2)	[-100,100]	-450
f_{38}	S_Schwefel_1_2_Noisy_CEC05(F_4)	[-100,100]	-450
f_{39}	RS_Elliptic_CEC05(F_3)	[-100,100]	-450
f_{40}	Com_CEC05(F_{15})	[-5,5]	120
f_{41}	H_Com_CEC05(F_{16})	[-5,5]	120
f_{42}	H_Com_Noisy_CEC05(F_{17})	[-5,5]	120
f_{43}	RH_Com_CEC05(F_{18})	[-5,5]	10
f_{44}	RH_Com_NarrowBasin_CEC05(F_{19})	[-5,5]	10
f_{45}	RH_Com_Bound_CEC05(F_{20})	[-5,5]	10

TABLE IV
PARAMETERS SETTINGS FOR f_3, f_{12}, f_{13}, f_{14}, AND ROTATED AND ROTATED SHIFTED FUNCTIONS

f	Parameter values
f_3	$y_i = \begin{cases} x_i, & \lvert x_i \rvert < 1/2, \\ round(2x_i), & \lvert x_i \rvert \geq 1/2 \end{cases}$
f_{12}	$u(x, a, k, m) = \begin{cases} k(x-a)^m, & x > a, \\ 0, & -a \leq x \leq a, \\ k(-x-a)^m, & x < -a. \end{cases}$
f_{13} f_{14}	m=10, $\mathbf{M}_{1-10}(f_{13})$=identity matrix, $c_{1-10}(f_{14}) = 2$ g_{1-2} = Sphere function, g_{3-4}=Rastrigin function g_{5-6}=Weierstrass function, g_{7-8}=Griewank function g_{9-10}=Ackley function $bias_k = 100(k-1), k = 1, 2, \ldots, 10$
f_{29}	$c = 100$
f_{23}–f_{28}, f_{30}	$c = 2$

TABLE V
CONFIGURATION OF THE INVOLVED PSO ALGORITHMS

Algorithm	Year	Population Topology	Parameter Settings
SPSO[28]	2007	Local ring	$\omega = 0.721$, $\eta_1 = \eta_2 = 1.193$
CPSO-H$_k$[44]	2004	Cooperative multi-swarm approach	ω: [0.4, 0.9], $\eta_1 = \eta_2 = 1.49$ k=6
FIPS[29]	2004	Local URing	$\chi = 0.7298$, $\sum c_i = 4.1$
CLPSO[25]	2006	Comprehensive learning	ω: [0.4, 0.9], $\eta = 2.0$
APSO[52]	2009	Global star	ω: [0.4, 0.9], $\eta_1 + \eta_2$: [3.0, 4.0] with adaptive tuning
TRIBES-D[8]	2009	Adaptive	
FPSO[11]	2009	Adaptive	ω: [0.4, 0.9], $\sum c_i = 4.0$
JADE[53]	2009	An adaptive DE algorithm	p=0.05, c=0.1
HRCGA[13]	2008	A real-coded GA	P_G=25%, N_F^G=200, N_M^G=400, N_F^L=5, N_M^L=100
G-CMA-ES[2]	2005	An ES algorithm	Default settings in [2]
APrMF[30]	2009	A memetic algorithm	Default settings in [30]

peer algorithms that are taken from the literature to compare with SLPSO in this paper.

The first algorithm is the cooperative PSO (CPSO-H$_k$) [44], which is a cooperative PSO model combined with the original PSO. For this algorithm, we also use the same value for the "*split factor*" $k = 6$ as it was used in the original paper [44]. The second algorithm is the fully informed PSO (FIPS) [29] with a U-ring topology that achieved the highest success rate. The third algorithm is the comprehensive learning PSO (CLSPO) [25], which was proposed for solving multi-modal problems and shows a good performance in comparison with eight other PSO algorithms. The fourth peer algorithm is the adaptive PSO (APSO) [52], which adaptively tunes the values of η_1, η_2, and ω based on the population distribution in the fitness landscape to achieve different purposes, such as exploration, exploitation, jumping out, and convergence. The fifth peer algorithm is the standard PSO (SPSO) where the implementation can be downloaded at [28] (note that, this is a bit different from the one proposed in [5] and more widely used than the one in [5]). We chose this algorithm to investigate how SLPSO performs compared to the standard version of PSO. The sixth PSO algorithm is Frankenstein's PSO (FPSO) [11], which is a composite PSO algorithm that combines a time-varying population topology, FIPS's velocity

update mechanism, and a decreasing inertial weight. Since the number of total fitness evaluations is fixed for the experimental study in this paper, we used the suggestion by [11] to set the parameters in FPSO where a slow topology change schedule and an intermediate inertia weight schedule were used in this paper. The seventh peer algorithm, the TRIBES PSO algorithm [8], is also an adaptive PSO algorithm which can adaptively tune the number of particles needed and the population topology. The implementation of TRIBES PSO is provided at [28]. Note that the implementation of the TRIBES algorithm used in this paper is a simplified version of the paper [8].

Four non-PSO algorithms are also included in the comparison with SLPSO. They are the JADE [53], HRCGA [13], APrMF [30], and G-CMA-ES [2] algorithms. JADE is an adaptive differential evolution algorithm with an optional external archive and HRCGA is a real-coded GA based on parent-centric crossover operators. We used the JADE with an archive in this paper since it has shown promising results compared with JADE without an archive in [53]. The parameters p for the DE/current-to-$p$best strategy and c for JADE were set to 0.05 and 0.1, respectively, as suggested in [53]. For the HRCGA algorithm, a combination of the global and local models were used where the number of female and male individuals for the two models were set to 200, 400 and 5, 100, respectively and the hybridization factor P_G was set to 25%. APrMF [30] is a probabilistic memetic algorithm, which is able to analyze the probability of evolution or individual learning. By adaptively choosing one of the two actions, the APrMF algorithm can accelerate the search of global optimum. G-CMA-ES [2] is a covariance matrix adaptation (CMA) evolution strategy (ES) algorithm with re-start mechanism and increasing population size. For SLPSO, η was set to 1.496, V_{max} was set to half of the search domain for each test function, which can be seen from Table I and Table III, and the default parameter settings suggested in Section V-C (to be introduced in the following section) were used for all problems unless explicitly stated in this paper.

To fairly compare SLPSO with the other 11 algorithms, all algorithms were implemented and run independently 30 times on the 45 test problems. The initial population and stop criteria were the same for all algorithms for each run. The maximal number of fitness evaluations (T_Fes) was used as the stop criteria for all algorithms on each function. The pair of swarm size and T_Fes was set to (10, 50000), (20, 100000), (30, 300000), and (100, 500000) for dimensions 10, 30, 50, and 100, respectively. For TRIBES-D, the adaptive swarm size model was used where it begins with a single particle and adaptively decreases or increases the number of particles that are needed. The population size for the HRCGA algorithm and the APrMF algorithm was suggested by [13] and [30], respectively. Any other parameter settings of the 11 peer algorithms are based on their optimal configurations as suggested by the corresponding papers.

C. Performance Metrics

1) Mean Values: We record the mean value of the difference between the best result found by the algorithms and the global optimum value over 30 runs on each problem, defined as:

$$mean = \sum_{k=1}^{30} (f(\vec{x}) - f(\vec{x}^*))/30 \tag{13}$$

where x and $\vec{x}^*$ represent the best solution found by an algorithm and the global optimum, respectively.

2) t-Test Comparison: To compare the performance of two algorithms at the statistical level, the two-tailed t-test with 58 degrees of freedom at a 0.05 level of significance was conducted between SLPSO and the peer algorithms. The performance difference is significant between two algorithms if the absolute value of the t-test result is greater than 2.0.

3) Success Rate: Another performance metric is the success rate, which is the ratio of the number of successful runs over the total number of runs. A successful run means the algorithm achieves the fixed accuracy level within the T_Fes fitness evaluations for a particular problem. The accuracy level for each test problem is given in Table VI. The accuracy levels for functions f_{31}-f_{45} are obtained from [41] and the accuracy levels of other functions are the same as used in [41], which were set according to the problem difficulty for all involved algorithms.

TABLE VI
ACCURACY LEVEL OF THE 45 PROBLEMS

Accuracy level	Function
1.0e-6	f_1, f_7, f_9, f_{11}, f_{12}, f_{16}, f_{18}, f_{19} f_{21}, f_{23}, f_{26}, f_{27}, f_{29}, f_{31}, f_{35}, f_{39}
0.01	f_2, f_3, f_4, f_5, f_6, f_8, f_{10}, f_{15}, f_{17}, f_{20}, f_{22} f_{24}, f_{25}, f_{28}, f_{30}, f_{32}, f_{33}, f_{34}, f_{36}, f_{37}, f_{38}
0.1	f_{13}, f_{14}, f_{40}, f_{41}, f_{42}, f_{43}, f_{44}, f_{45}

V. EXPERIMENTAL STUDY ON SLPSO

A. Self-Learning Mechanism Test

In order to investigate the level of effectiveness that the adaptive learning mechanism can bring to SLPSO, we carried out experiments on SLPSO with the adaptive learning mechanism and SLPSO without the adaptive learning mechanism (denoted Non-SLPSO) over all the test functions in 30 dimensions. For SLPSO, the default parameter settings were used. In Non-SLPSO, all the four operators have the same selection ratio of 0.25 during the whole evolutionary process. Table VII presents the results of SLPSO and Non-SLPSO on the 45 test functions.

Table VII shows that the result of SLPSO is much better than that of Non-SLPSO on 36 out of the 45 test problems. The reason to the result that SLPSO performs worse than Non-SLPSO on some problems is because the parameter values used were improper for those problems. This can be seen from the comparison of SLPSO with the default configurations and the optimal configurations in the next section where the performance of SLPSO with the optimal configuration is greatly improved. The comparison results show that the adaptive mechanism works well on most problems. It is also confirmed that different problems need different strategies to solve. To achieve the best performance, it is necessary to tune the selection ratios of the four operators so that the best strategy can effectively play its role.

TABLE VII
COMPARISON WITH RANDOM SELECTION FOR THE FOUR OPERATORS REGARDING THE MEAN VALUE

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Non-SLPSO	3.91e-13	6.84e-10	7.55e-07	1.52e-04	0.0201	3.82e-04	1.06e-07	22.8	1.73e-07
SLPSO	2.78e-50	0	0	4.50e-15	0.0227	3.82e-04	3.47e-14	2.06	1.35e-26
f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
Non-SLPSO	43.3	1.20e-04	4.11e-15	138	74.1	3.83e-04	0.00108	2.34e-05	4.39e-05
SLPSO	0.00793	0.00254	1.57e-32	59.8	33.3	3.82e-04	3.36e-14	0	0
f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
Non-SLPSO	4.11e-04	4.70e-04	0.0142	0.0897	1.12e-12	211	5.98e+03	3.79	1.95e-04
SLPSO	4.04e-04	4.25e-04	0.0144	0.0678	7.67e-50	134	3.96e+03	3.4	0
f	f_{28}	f_{29}	f_{30}	f_{31}	f_{32}	f_{33}	f_{34}	f_{35}	f_{36}
Non-SLPSO	5.13e+03	20.8	176	4.04e-05	2.17e-05	174	34.2	20.8	58.9
SLPSO	4.67e+03	20.5	111	1.00e-13	1.17e-13	115	32.9	20.4	10.7
f	f_{37}	f_{38}	f_{39}	f_{40}	f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
Non-SLPSO	234	1.16e+04	0.482	304	435	543	902	405	310
SLPSO	1.40e-06	1.16e+04	1.19e-13	305	420	914	1.56e+03	457	300

TABLE VIII
SELECTED EXAMPLES OF EFFECT ON THE THREE PARAMETERS (U_f,P_l, AND M) IN SLPSO

U_f	f_1	f_6	f_{20}	P_l	f_1	f_6	f_{20}	M	f_1	f_6	f_{20}
1	2.92e-32	7.9	35.5	0.05	1.74e-110	347	320	0	0.00817	0.295	0.138
3	1.41e-29	11.8	15.8	0.1	4.39e-128	205	138	0.1	0.00278	0.015	0.0176
7	1.10e-20	3.82e-04	11.8	0.3	5.01e-61	43.4	27.6	0.3	7.73e-04	0.0065	0.0076
10	3.81e-19	7.9	7.9	0.7	1.35e-26	3.95	11.8	0.7	7.83e-06	7.1e-04	0.0012
				1	3.81e-19	7.9	7.9	1	3.81e-19	7.9	7.9

B. Parameter Sensitivity Analysis of SLPSO

There are three key parameters in SLPSO: the update frequency (U_f), the learning probability (P_l), and the number of particles that learn from the $abest$ position (M). To find out how the three key parameters affect the performance of SLPSO, an experiment on the parameter sensitivity analysis of SLPSO was also conducted on all the 45 problems in 30 dimensions. The parameter of the number of particles that learn from the $abest$ position (M) is replaced by the percentage of the population size in this section. The default values of the three parameters were set to 10, 1.0, and 1.0, respectively. To separately test the effect of a particular parameter, we used the default values of the other two parameters. For example, to test the effect of U_f, we use a set of values for U_f and the default values for P_l ($P_l = 1.0$) and M ($M = 1.0$), respectively.

Three groups of experiments with U_f set to [1, 3, 7, 10], P_l set to [0.05, 0.1, 0.3, 0.7, 1.0], and M set to [0.0, 0.1, 0.3, 0.7, 1.0] were carried out separately. Table VIII shows the results on three example functions f_1, f_6, and f_{20}. Because of the space limitation, the results of the other functions are not provided in this paper. From Table VIII, similar observation of the effect on the three parameters can be seen that the optimal value of each parameter for a specific problem does depend on the property of that problem.

Although we have an insight on how the three parameters affect the performance of SLPSO, the corresponding optimal value of each parameter above may not be the real optimal parameter settings for a particular problem. It is difficult to obtain the real optimal parameter settings for a general problem for several reasons. First, we cannot test all the possible values of the three parameters as one of them is continuous, e.g., P_l. The value of M is population size dependent. Second, there may be relationships among the three parameters, i.e., they are not independent for SLPSO to achieve the best performance on a general problem.

In order to test whether the three key parameters of SLPSO are interdependent or not, further experiments were carried out based on the combination of the values given above for all the three parameters. The experimental results regarding the best results on each problem are summarized in Table IX and the corresponding optimal combinations of the three parameters for each problem are shown in Table X. It should be noted that we take the parameters of the optimal combinations as the real optimal configurations of SLPSO for each test problem even if they may not be the real optimal configurations.

TABLE IX
COMPARISON OF SLPSO WITH OPTIMAL AND DEFAULT CONFIGURATIONS IN TERMS OF MEAN VALUES

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Optimal	6.16e-139	0	0	0	0.00173	3.82e-04	1.06e-14	0.0551	3.36e-71
Default	2.78e-50	0	0	4.50e-15	0.0227	3.82e-04	3.47e-14	2.06	1.35e-26
f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
Optimal	7.45e-10	1.89e-05	1.57e-32	27	33.7	3.82e-04	1.23e-14	0	0
Default	0.00793	0.00254	1.57e-32	59.8	33.3	3.82e-04	3.36e-14	0	0
f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
Optimal	1.48e-04	4.02e-04	0.00941	0.0298	3.22e-111	88.4	2.49e+03	0.249	0
Default	4.04e-04	4.25e-04	0.0144	0.0678	7.67e-50	134	3.96e+03	3.4	0
f	f_{28}	f_{29}	f_{30}	f_{31}	f_{32}	f_{33}	f_{34}	f_{35}	f_{36}
Optimal	1.98e+03	20.4	81.1	5.49e-14	9.28e-14	75.4	28.7	20.3	1.22
Default	4.67e+03	20.5	111	1.00e-13	1.17e-13	115	32.9	20.4	10.7
f	f_{37}	f_{38}	f_{39}	f_{40}	f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
Optimal	4.86e-10	1.71e+03	5.68e-14	265	403	488	400	363	290
Default	1.40e-06	1.16e+04	1.19e-13	305	420	914	1.56e+03	457	300

TABLE X
OPTIMAL PARAMETER SETTINGS FOR THE 45 PROBLEMS

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
U_f	3	1	1	1	3	1	1	1	3	1	10	1	1	1	1
P_l	0.1	0.05	0.1	0.05	0.05	0.05	0.05	0.1	0.1	0.1	0.7	0.3	0.3	0.1	0.05
M	1	0.7	0.7	0.7	0	0.1	0.7	1	1	1	1	1	0.1	0.3	0.1
f	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}	f_{29}	f_{30}
U_f	7	1	1	7	1	1	1	7	10	3	1	1	3	1	7
P_l	0.05	0.7	0.1	0.05	0.05	0.05	0.05	0.1	0.05	0.05	0.05	0.3	0.05	0.05	0.05
M	1	1	1	0	0.1	0	0.7	1	1	0	0	1	0.1	1	0
f	f_{31}	f_{32}	f_{33}	f_{34}	f_{35}	f_{36}	f_{37}	f_{38}	f_{39}	f_{40}	f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
U_f	10	7	10	1	1	10	3	10	7	3	3	1	3	1	10
P_l	0.05	0.3	0.05	0.05	0.05	0.1	0.1	0.05	0.05	0.05	0.1	0.1	0.05	0.1	0.3
M	1	1	0	1	1	1	1	0	1	0.3	0	0.1	0.3	0.3	1

Comparing the optimal configurations for each problem obtained in this section with the results in the above section, we can see that the three key parameters do have interrelationship. Taking the Sphere function (f_1) as an example, the optimal combination of the three parameters are 3, 0.1, and 1 for U_f, P_l, and M, respectively, which are different from the above experimental results where the corresponding optimal values are 1, 0.05, and 1, respectively. Therefore, there is probably no effective general rule that can be applied to set up the three parameters in SLPSO.

C. Parameter Tuning in SLPSO

The values of these three key parameters significantly affect the performance of SLPSO on most problems tested in this paper. To achieve the best performance for SLPSO, different optimal values of U_f, P_l, and M are needed on different problems, which can be seen from the above experimental study on the parameters sensitivity analysis. The aim of this section is to suggest several approaches to adjusting the values of the three parameters for a general problem without manually tuning the parameters.

First, we present a general solution to set up the values of the update frequency (U_f) for SLPSO on all problems.

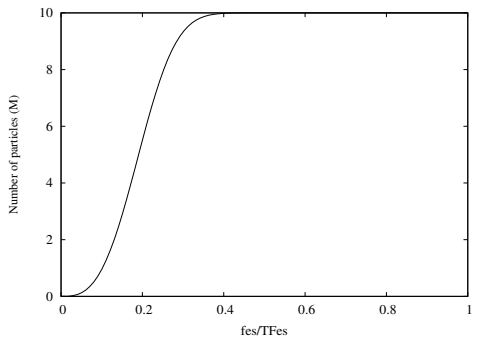

Fig. 1. The number of particles that use the convergence operator at different iterations in a swarm of 10 particles.

Based on our previous work in [21], the optimal values of U_f mainly distribute from 1 to 10 on most test problems in [21]. This information is very useful for SLPSO to set up U_f for a general problem. In order to use this information, we assign each particle with a different value of U_f instead of using the same value of U_f for all particles. The value of U_f for particle k is defined as follows:

$$U_f^k = \max(10 * exp(-(1.6 \cdot k/N)^4), 1) \tag{14}$$

where N is the population size and U_f^k is the update frequency of particle k. By this scheme, the values of U_f of all particles distribute from 1 to 10. As a result, it is possible that some particles may be able to achieve the optimal value of U_f for different problems. Similar idea was implemented in CLPSO [25] to adjust the learning probability P_c for each particle on a general problem.

For the learning probability (P_l), we use the same setup method as used for the update frequency where the learning probabilities of all particles distribute between 0.05 and 1.0, which is described as follows:

$$P_l^k = \max(1 - exp(-(1.6 \cdot k/N)^4), 0.05) \tag{15}$$

In order to reduce the risk of using improper values of U_f and P_l for a particular particle, we generate a permutation of index numbers of particles every iteration and then update the values of U_f and P_l for each particle. As to how many particles should use the convergence operator, we use the following formula:

$$M(fes) = N \cdot (1 - exp(-100(fes/T\_Fes)^3)) \tag{16}$$

where T_Fes is the total number of fitness evaluations allowed for a run. Fig. 1 shows the function relationship between M and fes. From Fig. 1, it can be seen that all particles initially do not use the convergence operator in order to focus on local search. However, to accelerate the convergence, the number of particles that use the convergence operator will gradually increase to the maximum value of 10 when the number of fitness evaluations reaches 40% of the total number of fitness evaluations. The main reason of using this particular method instead of using the selection ratio of the convergence operator to tune the value of M lies in that we need to bring another new parameter of the maximum selection ratio for the convergence operator in order to achieve the objective. In addition, the optimal value of the maximum selection ratio for different problems probably is different and unknown.

Algorithm 5 UpdatePar()

1: Create a permutation of index number;
2: Update U_f for each particle by Eq. (14);
3: Update P_l for each particle by Eq. (15);
4: Calculate the number of particles using the convergence operator by Eq. (16);
5: Update related information of the four operators for each particle by Algorithm 3;
6: Calculate the inertia weight ω by Eq. (17);

In SLPSO, the inertia weight ω linearly decreases from 0.9 to 0.4 according to the following equation:

$$\omega(fes) = 0.9 - 0.5 * fes/T\_Fes \tag{17}$$

In order to show the effectiveness of these parameter tuning methods, comparison between the optimal configurations and the default configurations suggested in this section was conducted. The results are shown in Table IX. From the results, it can be seen that the performance of SLPSO with the default configurations is comparable to that with the optimal configurations on most test problems. However, this result also shows that it is necessary to further study how to effectively set up the parameters of SLPSO for general problems.

The parameter tuning equations used in SLPSO were developed empirically from our experimental study based on the problems selected in this paper. Although these equations and the constants used in them, e.g., 1.6 in Eq.(14) and Eq.(15), may be not the optimal ones to set the values for the parameters of SLPSO, they were not randomly chosen but a result of a careful analysis. The analysis of all these equations is not discussed as it is not the main task of this paper. Here, we just provide some ideas of how to tune the parameters for SLPSO for general problems, and users can design their own methods to set the parameters for SLPSO. However, we would like to discuss some experience from our experimental study in terms of tuning the three key parameters in SLPSO. For the parameters U_f and P_l, we should allow enough particles to use the extreme values, e.g., 25% of total particles using the value of 0.05 or 1.0 for P_l. Regarding M, it is important that no particle learns from the $abest$ position initially and the number of particles that learn from the $abest$ position gradually increases until all particles learn from the $abest$ position when the number of fitness evaluations reaches about 40% of total fitness evaluations. Although the parameter tuning methods were developed based on the problems used in this paper, they can also be used for new problems. These methods work well and the evidence can be seen from the comparison of SLPSO that uses these methods to set the parameters with other algorithms in Section VI. The update operations of these parameters in SLPSO are summarized in Algorithm 5.

D. Selection Ratios of the Four Operators

So far, although we have recognized that the adaptive learning mechanism is beneficial, it is not yet clear how the four operators in SLPSO would perform on a general problem. To answer this question, we analyze the behavior of each learning operator on some selected problems of 30 dimensions over 30 runs. To clearly show the learning behavior of the

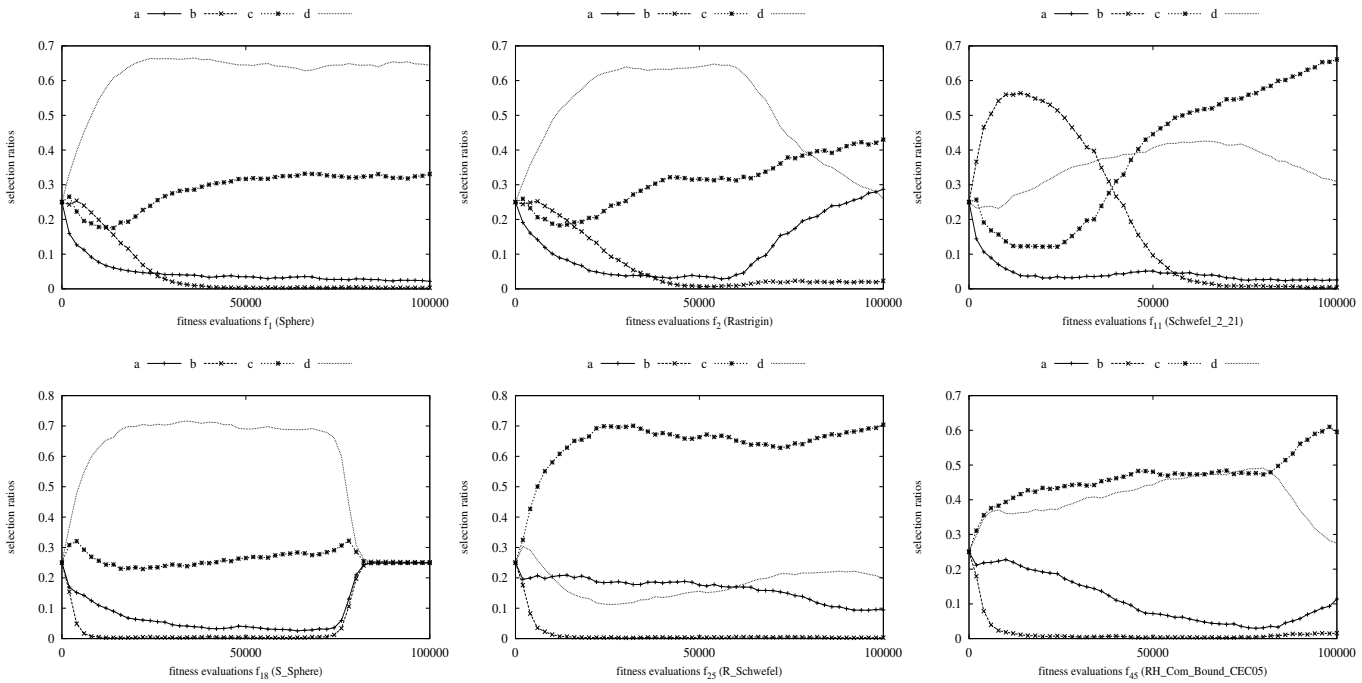

Fig. 2. Selection ratios of the four operators on six selected functions, where a, b, c, and d denote the exploitation, jumping-out, exploration, and convergence operators, respectively.

four operators without impact from other factors, we allowed all particles to use the convergence learning operator through the whole run in this set of experiments. Fig. 2 only presents the results on six problems since similar observations can be obtained on other functions. From Fig. 2, several observations can be made and are described below.

First, the operators have very different performance on each specific problem and their performances also vary on different problems. On many problems, the best learning operator changes from the beginning to the end of evolution. For example, the best learning operator is the jumping-out operator (operator b) for the Schwefel_2_21 function (f_{11}) at the early stage. However, its selection ratio decreases after $fes = 20000$ until the end. On the contrast, the selection ratio of the exploration operator (operator c) increases and the exploration operator becomes the best learning operator after about 50000 evaluations. There is a period from $fes = 40000$ to $fes = 50000$ where the convergence operator (operator d) turns out to be the best learning operator. The exploitation operator (operator a) may be not suitable for the Schwefel_2_21 function as its selection ratio decreases from the beginning and remains at a very low level till the end.

Second, for some functions, the best learning operator does not change during the whole evolution process, e.g., the convergence operator on the Sphere function (f_1) and the exploration operator on the R_Schwefel function (f_{25}). The corresponding selection ratios remain at the highest level during the whole evolution process on these two functions.

In addition, the convergence status appears at the population level for the S_Sphere function (f_{18}). From the graph of function f_{18}, we can see that the selection ratios of the four operators return to the initial state, where they almost have the same value of 0.25. The convergence status shows only if none of the four learning operators can help particles to move to better areas. Of course, when a whole swarm converges to the global optimum, this phenomenon will show.

In general, we can get the following observations: 1) due to the advantages of the convergence operator discussed in Section III-E, particles get the greatest benefit from the convergence operator on functions f_1, f_2, and f_{18}; 2) although the jumping-out operator has the lowest selection ratio on most problems, it does help the search during a certain period on some problems, e.g., f_{11}; 3) particles always get benefit from the exploration operator and even the largest benefit from it on some functions, e.g., f_{25}, and f_{45}; 4) the exploitation operator may work for a short period after particles jump into a new local area as its selection ratio never reaches the highest level.

The results of this section confirm that different problems need different kinds of intelligence to solve them and that an adaptive method is needed to automatically switch to an appropriate learning operator at each evolutionary stage. We can also draw the conclusion that the individual level of intelligence works well for most problems.

VI. Experimental Study on Comparison with Other Algorithms

A. Comparison Regarding Mean and Variance Values

In this section, experiments were conducted to compare SLPSO with 11 peer algorithms described in Section IV-B. Each algorithm was executed 30 independent runs over the 45 test problems in four dimensional cases, which are 10, 30, 50, and 100 dimensions, respectively, except the APrMF algorithm in 100 dimensions. The reason is that the program of APrMF

TABLE XI
COMPARISON RESULTS OF MEANS AND VARIANCES IN 30 DIMENSIONS

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
SLPSO	2.78e-50±8.09e-49	**0**±0	**0**±0	4.50e-15±3.55e-14	0.0227±0.144	**3.82e-04**±7.31e-12	3.47e-14±4.45e-14	2.06±12.3	1.35e-26±3.88e-25
APSO	1.64e-51±3.75e-50	5.7±11.9	2.3±12.3	0.27±1.73	0.013±0.0863	686±9.83e+03	0.0771±1.58	23.4±70.5	4.56e-32±8.16e-31
CLPSO	3.84e-13±1.51e-12	2.36e-04±0.00129	0.302±3.5	1.67e-06±7.52e-06	2.53e-08±3.47e-07	39.5±306	4.05e-07±9.96e-07	22.9±10.9	1.09e-08±2.57e-08
CPSOH	1.64e-30±2.72e-29	10.8±34.7	10.9±35.7	1.57e-07±3.00e-06	0.0235±0.0918	2.67e+03±3.89e+03	2.91e-14±4.02e-14	18.4±8.13	1.72e-14±2.95e-13
FIPS	5.36e-12±2.28e-11	75±83.2	82.4±86.8	2.06e-04±3.20e-04	0.00162±0.0368	2.46e+03±4.43e+03	5.23e-07±6.95e-07	24.7±1.53	1.17e-07±2.33e-07
SPSO	1.15e-69±2.86e-68	46.1±78.5	48±70.9	2.31±7.99	0.0144±0.108	3.79e+03±4.21e+03	1.35±5.67	17.4±9.25	7.24e-37±1.79e-35
JADE	2.11e-104±5.84e-103	0.398±3.33	2.53±7.45	0.175±1.84	0.0072±0.0772	351±925	0.122±2.01	0.399±6.55	8.24e-57±1.16e-55
HRCGA	7.05e-105±2.08e-103	45.5±167	77.2±174	0.294±0.97	0.00812±0.061	1.77e+03±5.19e+03	4.62e-14±1.64e-13	7.38±10.6	4.42e-60±6.58e-59
FPSO	2.71e-04±6.35e-04	129±68.2	115±108	0.0429±0.0492	0.11±0.582	2.94e+03±4.52e+03	0.0037±0.00628	26.1±1.37	5.28e-04±8.13e-04
TRIBES-D	**0**±0	**0**±0	**0**±0	**0**±0	**0**±0	5.16e+03±711	**0**±0	22.1±0.286	**0**±0
APrMF	**0**±0	**0**±0	13.1±3.28	4.41±2.2	3.81e-05±1.72e-04	3.09e+03±349	3.32e-08±9.77e-08	50.4±24.7	0.0029±0.00282
G-CMA-ES	2.78e-15±3.94e-31	12.9±7.11e-15	20±0	**0**±0	4.66e-15±0	1.24e+04±3.64e-12	3.43e-11±0	**4.79e-15**±2.37e-30	1.87e-10±7.75e-26

f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
SLPSO	0.00793±0.233	0.00254±0.0186	**1.57e-32**±4.50e-47	59.8±505	33.3±258	**3.82e-04**±8.66e-12	3.36e-14±4.00e-14	**0**±0	**0**±0
APSO	0.00991±0.0503	0.203±0.909	0.0173±0.212	73.3±599	510±3.82e+03	318±4.25e+03	1.07±18.9	5.58±12.4	1.14e-26±1.36e-25
CLPSO	1.48e+03±1.83e+03	8.69±6.83	4.47e-14±2.00e-13	40.1±255	75.2±605	15±210	4.77e-05±5.09e-04	0.465±3.91	1.28e-12±4.25e-12
CPSOH	309±6.73e+03	1.69e-04±0.0016	4.61e-32±6.45e-31	264±1.1e+03	794±3.72e+03	2.27e+03±3.06e+03	0.36±3.77	25±49.7	1.89±13.5
FIPS	321±847	0.0541±0.129	2.27e-13±1.18e-12	306±1.32e+03	166±1.17e+03	2.37e+03±5.02e+03	8.35e-06±1.83e-04	50.4±76	14.6±358
SPSO	4.54e-04±0.00431	1.41±4.41	0.287±2.47	72±942	93.7±1.29e+03	3.19e+03±4.11e+03	1.1±4.88	58.2±73.5	1.01e-28±7.08e-28
JADE	8.71e-12±1.51e-10	1.15e-04±8.82e-04	0.0904±1.72	30±251	173±1.31e+03	377±791	0.152±2.25	1.2±5.34	6.73e-30±1.99e-28
HRCGA	349±9.87e+03	0.00775±0.0491	2.23e-31±5.52e-30	**26.7**±242	23.3±232	1.93e+03±2.91e+03	**1.77e-14**±5.94e-14	45±191	2.72e-28±2.17e-27
FPSO	2.49e+03±2.96e+03	3.32±4.14	9.13e-05±0.00168	516±1.26e+03	419±1.76e+03	2.36e+03±4.62e+03	1.01±14.2	83.3±64	316±6.69e+03
TRIBES-D	**0**±0	**0**±0	2.39e-04±2.18e-04	96.7±143	395±621	5.06e+03±574	1.05±0.866	90.9±30.1	1.08e-27±2.19e-28
APrMF	2.62e+04±7.99e+03	61.1±12.2	0.0276±-1	378±125	332±45.6	2.93e+03±341	0.0121±0.0415	4.2±2.94	0.0416±0.184
G-CMA-ES	7.97e-15±3.16e-30	2.01e-11±9.69e-27	7.17e-15±4.73e-30	100±1.42e-14	**5.02e-08**±1.32e-23	1.22e+04±1.82e-12	19.6±1.07e-14	235±2.84e-14	2.52e-15±0

f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
SLPSO	4.04e-04±0.00138	**4.25e-04**±8.03e-05	0.0144±0.0209	0.0678±0.133	7.67e-50±9.99e-49	134±348	3.96e+03±1.09e+04	3.4±5.88	**0**±0
APSO	2.40e-04±7.25e-04	788±8.85e+03	0.0113±0.0146	5.72±12	4.38e-42±1.29e-40	113±169	7.58e+03±6.7e+03	3.76±5.7	1.05e-25±2.19e-24
CLPSO	2.22e-04±1.73e-04	23.7±309	0.0108±0.004	0.202±1.66	7.56e-05±4.32e-04	134±102	7.62e+03±3.1e+03	2.25±3.21	3.47e-04±0.00146
CPSOH	1.87e-04±2.49e-04	2.76e+03±2.47e+03	0.0101±0.00558	13.1±43.6	1.56e-28±1.97e-27	158±286	8.48e+03±5.54e+03	8.25±47.3	4.91±100
FIPS	1.49e-04±8.88e-05	2.56e+03±4.3e+03	0.00891±0.00263	79.3±91.6	2.29e-10±8.54e-10	184±58.7	5.88e+03±5.54e+03	8.61e-06±2.42e-05	338±5.82e+03
SPSO	1.49e-04±1.66e-04	3.48e+03±3.45e+03	0.966±4.47	49.4±55.8	1.10e-56±2.61e-55	84.9±257	5.33e+03±4.59e+03	1.61±4.27	9.56e-28±7.61e-27
JADE	1.65e-04±5.83e-04	493±1.22e+03	0.266±2.6	0.859±5.82	1.66e-88±4.76e-87	50.7±58.5	5.53e+03±4.01e+03	0.742±4.35	5.11e-29±6.97e-28
HRCGA	5.98e-04±0.00653	1.97e+03±3.36e+03	0.0149±0.0388	66.8±250	2.59e-83±7.08e-82	149±420	4.1e+03±6.2e+03	0.529±3.42	1.66e-27±8.12e-27
FPSO	6.35e-04±0.00104	3.04e+03±3.74e+03	0.0132±0.0086	123±95.2	0.00368±0.0129	208±59.8	6e+03±9.5e+03	0.04±0.0819	670±5.86e+03
TRIBES-D	**4.65e-09**±3.48e-09	4.95e+03±660	**5.25e-08**±6.95e-08	**1.27e-06**±1.86e-06	**0**±0	**0**±0	7.13e+03±1.03e+03	**0**±0	3.27e-15±1.67e-14
APrMF	71.1±131	1.09e+04±42.3	20.4±0.49	437±128	1.92e+03±2.38e+03	253±37.9	**624**±687	18.2±0.455	2.54e+03±2.79e+03
G-CMA-ES	1.02e-04±1.09e-05	1.24e+04±10.1	0.00763±4.30e-04	20.1±7.11	1.80e-15±7.89e-31	20.9±1.78e-14	6.83e+03±2.73e-12	5.18e-11±0	2.98e-15±1.18e-30

f	f_{28}	f_{29}	f_{30}	f_{31}	f_{32}	f_{33}	f_{34}	f_{35}	f_{36}
SLPSO	4.67e+03±1.29e+04	**20.5**±0.906	111±224	1.00e-13±1.74e-13	**1.17e-13**±1.79e-13	115±249	32.9±14.5	20.4±0.846	10.7±140
APSO	7.85e+03±6.02e+03	21.2±0.579	152±215	7.20e-14±1.38e-13	5.74±9.09	133±300	30.4±25.1	21.2±0.357	29.9±168
CLPSO	7.88e+03±2.72e+03	21±0.208	150±133	8.01e-13±3.40e-12	0.199±2.59	142±84.5	30.1±7.76	21±0.191	10.4±56.7
CPSOH	8.45e+03±3.95e+03	20.9±0.39	245±242	0.0426±1.25	35.5±58.8	221±287	28.2±17.4	20.9±0.373	1.73e+03±4.23e+04
FIPS	5.98e+03±5.71e+03	21±0.267	180±49.1	130±2.47e+03	48.1±57.1	190±60.4	37.5±10.3	21±0.268	3.56e+06±8.31e+07
SPSO	5.79e+03±4.46e+03	20.9±0.409	80.7±185	1.63e-13±4.00e-13	47.9±75.1	82.4±200	32.5±27.4	21±0.35	29.1±214
JADE	5.96e+03±4.19e+03	20.9±0.315	63.7±74.6	5.68e-14±8.04e-14	0.73±4.21	58.1±85.4	29±7.6	20.9±0.308	0.532±7.42
HRCGA	4.53e+03±5.11e+03	21.1±0.318	125±415	2.10e-13±7.29e-13	29.9±48.7	107±407	16.7±43.7	21.1±0.391	8.59±65
FPSO	6.13e+03±7.2e+03	21±0.256	196±61.1	447±4.23e+03	78.9±75.1	200±80.8	40.2±6.69	21±0.273	3.71e+07±5.80e+08
TRIBES-D	7.45e+03±1.19e+03	21±0.0458	104±38.3	**5.12e-14**±1.71e-14	103±37.8	107±46.8	26.5±4.34	21±0.0444	75.2±143
APrMF	**917**±724	20.6±0.183	232±32.3	0.755±4.06	3.92±4.42	231±37.1	30.9±2.07	20.5±0.16	5.57e+10±1.92e+03
G-CMA-ES	8.01e+03±5.46e-12	21±3.55e-15	**44.8**±0	5.68e-14±0	66.7±5.68e-14	**52.7**±3.55e-14	**0**±0	**20**±1.07e-14	**5.68e-14**±0

f	f_{37}	f_{38}	f_{39}	f_{40}	f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
SLPSO	1.40e-06±1.57e-05	1.16e+04±2.99e+04	1.19e-13±1.47e-13	305±837	420±410	914±1.5e+03	1.56e+03±2.84e+03	457±1.06e+03	300±1.01e-11
APSO	0.0182±0.0953	946±3.3e+03	7.39e-14±1.43e-13	363±714	462±1.39e+03	656±2.96e+03	1.71e+03±3.31e+03	312±939	300±2.81e-11
CLPSO	3.38e+03±3.19e+03	1.12e+04±1.57e+04	4.06e-09±1.27e-08	291±533	400±0.968	838±843	1.5e+03±2.43e+03	492±988	345±615
CPSOH	1.34e+03±1.14e+04	1.07e+04±7.25e+04	1.24±36.4	639±1.9e+03	1.22e+03±1.92e+03	1.35e+03±1.76e+03	2.08e+03±818	649±372	619±259
FIPS	172±444	770±2.81e+03	8.87e+03±2.62e+05	477±623	451±767	471±730	430±897	520±994	534±803
SPSO	0.616±17.7	540±6.03e+03	51±1.51e+03	418±1.09e+03	403±98.3	426±425	545±2.4e+03	520±1.12e+03	586±617
JADE	2.33e-09±6.59e-08	1.97e+03±1.14e+04	5.49e-14±5.59e-14	343±595	464±1.35e+03	459±1.38e+03	509±2.24e+03	417±1.17e+03	433±890
HRCGA	11.7±103	1.31e+03±3.62e+03	8.01e-13±4.99e-12	320±606	**400**±3.54e-11	465±371	**400**±7.21e-11	231±626	572±666
FPSO	1.67e+03±2.84e+03	2.49e+03±4.17e+03	5.96e+04±1.10e+06	459±1.42e+03	549±1.41e+03	988±445	837±3.17e+03	644±175	620±159
TRIBES-D	1.96±1.32	**339**±168	3.44e+06±1.22e+06	470±275	643±491	496±345	637±247	900±0	900±0
APrMF	4.21e+04±1.07e+04	1.40e+05±3.16e+04	**9.59e-18**±4.68e-17	583±93.2	404±102	**393**±30.2	1.26e+03±4.78	**1**±1	**1**±1
G-CMA-ES	**5.68e-14**±0	7.38e+04±9.15e+04	5.68e-14±0	**200**±5.68e-14	**400**±1.14e-13	1.5e+03±403	1.94e+03±0	659±0	400±1.71e-13

obtained from the authors (the MAPv1-win version [30] in Windows systems, e.g., Windows XP and Windows 7) crashed on most test problems used in this paper in 100-dimension. Therefore, we cannot provide the corresponding results on problems in 100-dimension. However, this does not affect so much the comparison as we have the results of APrMF in 10, 30, and 50 dimensions. The corresponding results are provided in Tables XI and XII, respectively, where the best result on each problem among all algorithms is shown in bold. Although the detailed results in 10 and 50 dimensional cases are not provided in this paper due to the space limit, we still discuss the results in this section. The first row in Table XIII shows the number of problems where the best result was achieved by each algorithm over the 45 problems in 10, 30, 50, and 100 dimensions, respectively.

From the first row in Table XIII, it can be seen that SLPSO achieves a slightly worse result than the best performer TRIBES-D among the 12 algorithms. The number of the best results achieved by SLPSO for the four dimensional cases are 11, 11, 12, and 14, which are slightly smaller than the figures (12, 17, 14, and 16) obtained by the TRIBES-D algorithm, respectively. G-CMA-ES and HRCGA achieve the third and fourth ranking, respectively. The performances of FIPS, CPSOH, and FPSO are the worst in terms of the number of best mean results among all the peer algorithms where the figures are all 0 for the four dimensional cases.

Among the 12 traditional problems, TRIBES-D achieves the best results where it found the global optima of nine

TABLE XII
COMPARISON RESULTS OF MEANS AND VARIANCES IN 100 DIMENSIONS

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
SLPSO	2.42e-38±3.83e-37	**0**±0	**0**±0	2.39e-13±3.97e-13	0.0126±0.084	**0.00127**±6.32e-11	1.72e-13±9.83e-14	52.6±192	1.50e-18±5.57e-18
APSO	1.82e+03±3.02e+04	140±1.22e+03	194±1.11e+03	2.87±63.5	18.7±351	9.2e+03±2.72e+04	1±20.6	113±163	32.7±267
CLPSO	3.79e-08±4.51e-08	61.5±34.4	79.8±26.8	0.00214±0.00113	9.50e-08±3.75e-07	2.91e+03±2.95e+03	3.78e-05±3.33e-05	94.3±6.42	8.91e-06±7.69e-06
CPSOH	7.68e-24±3.96e-23	183±223	177±294	0.135±2.03	0.00493±0.0522	1.65e+04±8.21e+03	3.69e-13±9.88e-13	151±1.17e+03	9.12e-13±3.05e-12
FIPS	1.4±1.66	696±128	725±207	2.11±0.623	0.576±0.414	2.3e+04±5.98e+03	0.196±0.124	97±1.73	0.159±0.0993
SPSO	2.68e-43±4.55e-42	209±220	113±189	11.2±22	0.0125±0.147	1.48e+04±9.66e+03	2.17±2.8	90.2±53.2	3.66e-24±7.24e-23
JADE	1.12e-104±1.52e-103	2.06±9.11	33.6±27.6	5.19±12.5	0.00335±0.0535	549±1.35e+03	1.37±1.71	0.535±7.42	6.28e-51±1.40e-49
HRCGA	4.30e-74±6.04e-73	177±179	267±214	7.4±7.48	0.0965±0.45	4.11e+03±3.36e+03	2.3±1.53	61.4±21.6	5.32e-33±1.57e-31
FPSO	4.05e+03±3.06e+03	688±142	706±141	46±8.9	37.5±27.6	2.53e+04±3.94e+03	8.42±2.1	267±143	41.5±10.7
TRIBES-D	**0**±0	**0**±0	**0**±0	**0**±0	**0**±0	1.79e+04±1.14e+03	**0**±0	91.8±0.185	**0**±0
G-CMA-ES	5.72e-16±0	65.7±0	92±0	1±1	1.65e-14±0	4.15e+04±7.28e-12	4.72e-11±0	**2.49e-14**±6.31e-30	2.90e-09±0
f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
SLPSO	7.56±29.9	0.15±0.302	**1.57e-32**±4.50e-47	46.7±612	105±1.77e+03	**-2.29e+04**±1.00e-10	**1.63e-13**±8.50e-14	**0**±0	**0**±0
APSO	5.33e+04±7.35e+05	56.9±96.4	6.98e+06±1.43e+08	371±2.71e+03	1.78e+03±2.89e+03	-1.12e+04±6.08e+04	17.3±28.8	429±2.23e+03	1.27e+04±1.85e+05
CLPSO	6.65e+04±3.42e+04	16±6.9	3.19e-09±5.40e-09	14.7±423	1.33e+03±2.81e+03	-2.24e+04±965	0.0375±0.582	67.2±32.2	3.16e-07±4.88e-07
CPSOH	3.75e+04±1.04e+05	0.00786±0.0184	1.04e-24±8.83e-24	618±350	1.47e+03±2.13e+03	3.05e+03±1.27e+04	5.6±16.9	447±317	672±2.91e+03
FIPS	1.15e+05±5.95e+04	25.1±9.07	34.5±61.6	105±535	81.3±482	-6.97e+03±1.6e+04	0.0907±0.0682	489±179	0.407±0.429
SPSO	2.96e+03±9.14e+03	36.2±24	0.698±5.62	65.3±1.35e+03	385±3.85e+03	-1.18e+03±1.66e+04	2.2±2.81	253±299	1.50e-27±3.39e-27
JADE	2.08±28.5	14.9±11.4	0.163±1.44	30±602	845±4.69e+03	-2.23e+04±1.28e+03	1.29±2.58	2.65±8.48	**0**±0
HRCGA	9.24e+03±2.02e+04	42.7±37.5	0.544±3.93	1.11e-05±3.27e-04	20±590	-1.57e+04±7.85e+03	1.03±3.28	130±155	3.59e-19±9.86e-18
FPSO	2.44e+04±1.73e+04	15.2±4.84	34.5±24.6	647±139	1.3e+03±2.66e+03	-3.09e+03±8.85e+03	8.64±2.3	782±179	4.2e+03±2.47e+03
TRIBES-D	**0**±0	**0**±0	1.41e-04±1.20e-04	206±449	977±896	6.48e+03±2.05e+03	2.37±0.41	468±95.5	5.03e-26±2.46e-26
G-CMA-ES	1.64e-13±2.52e-29	2.39e-10±1.03e-25	5.58e-15±2.37e-30	**6.83e-08**±3.97e-23	**8.57e-08**±3.97e-23	1.37e+04±7.28e-12	19.6±0	771±2.27e-13	1.30e-15±7.89e-31
f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
SLPSO	0.00861±0.0129	**0.00245**±0.00146	0.0414±0.0326	1.37±1.87	1.07e-35±2.19e-34	802±1.59e+03	2.8e+04±2.03e+04	17.5±5.58	**0**±0
APSO	0.00311±0.0048	1.1e+04±3.17e+04	0.483±13.5	115±1.05e+03	4.32e+03±9.63e+04	927±2.19e+03	4.05e+04±3.07e+04	17.2±33.7	2.43e+04±3.46e+05
CLPSO	0.0014±5.13e-04	2.98e+03±3.4e+03	0.0149±0.00298	73±31.7	0.00569±0.019	885±169	3.69e+04±4.35e+03	4.14±5.65	0.065±0.0948
CPSOH	0.00133±0.00134	1.64e+04±6.98e+03	0.0153±0.0077	215±248	3.77e-22±2.12e-21	961±640	3.59e+04±9.63e+03	15.6±43.8	3.48e+03±1.25e+04
FIPS	1.4±1.48	2.31e+04±7.08e+03	0.191±0.171	681±166	11.3±12.3	940±122	3.27e+04±2.85e+03	2.17±1.01	2.72±3.32
SPSO	0.00116±0.00171	1.32e+04±1e+04	2.01±3.05	202±206	3.91e-32±3.65e-31	351±643	2.45e+04±9.9e+03	2.78±2.5	9.19e-27±1.98e-26
JADE	0.00129±0.00213	711±1.41e+03	1.4±2.13	2.7±12	6.99e-83±8.94e-82	240±200	2.57e+04±6.57e+03	2.41±2.09	4.27e-28±3.73e-27
HRCGA	2.6±16.5	3.84e+03±2.99e+03	2.35±1.91	182±155	7.75e-32±2.15e-30	310±255	**2.02e+04**±1.06e+04	3.69±1.87	1.36e-17±2.56e-16
FPSO	4.11e+03±3.17e+03	2.53e+04±4.61e+03	8.19±2.12	681±182	9.18e+03±5.8e+03	807±115	3.43e+04±7.76e+03	10.9±2.51	1.09e+04±5.98e+03
TRIBES-D	**2.62e-09**±2.05e-09	1.75e+04±1.5e+03	**7.11e-09**±7.59e-09	**5.07e-07**±3.81e-07	**0**±0	**0**±0	3.08e+04±2.48e+03	**0**±0	3.21e-23±6.51e-23
G-CMA-ES	7.33e-04±5.54e-05	4.15e+04±6.69e-06	0.0112±4.27e-04	94.7±22.4	2.12e-15±3.94e-31	142±1.14e-13	2.5e+04±7.28e-12	6.01e-11±0	1.70e-15±3.94e-31
f	f_{28}	f_{29}	f_{30}	f_{31}	f_{32}	f_{33}	f_{34}	f_{35}	f_{36}
SLPSO	3.08e+04±1.44e+04	**20.3**±0.946	617±1.22e+03	6.16e-13±4.55e-13	**6.04e-13**±5.42e-13	617±1.22e+03	144±35.5	**20.4**±0.857	22.9±165
APSO	4.75e+04±3.47e+04	21.5±0.334	1.71e+03±3.24e+03	1.27e+04±1.85e+05	313±1.87e+03	1.71e+03±3.24e+03	145±83.4	21.5±0.348	2.50e+10±3.67e+11
CLPSO	4.07e+04±4.22e+03	21.3±0.127	1.01e+03±232	3.16e-07±4.88e-07	68.1±30.3	1.01e+03±232	141±14	21.4±0.122	332±376
CPSOH	3.99e+04±1.48e+04	21.3±0.158	1.21e+03±886	672±2.91e+03	440±289	1.21e+03±886	127±32.9	21.3±0.192	1.09e+07±5.61e+07
FIPS	3.35e+04±4.45e+03	21.3±0.181	908±125	0.407±0.429	491±128	908±125	160±7.54	21.3±0.16	6.67e+03±1.12e+04
SPSO	2.73e+04±8.88e+03	21.3±0.231	358±544	3.92e-13±1.34e-12	257±231	358±544	141±19.3	21.3±0.163	141±306
JADE	2.89e+04±7.65e+03	21.3±0.149	361±214	1.29e-13±1.79e-13	2.35±9.49	361±214	133±16.6	21.3±0.161	**0.532**±7.42
HRCGA	**2.35e+04**±6.58e+03	21.4±0.183	**257**±706	4.99e-10±6.40e-09	128±125	**257**±706	57.8±30.9	21.4±0.167	74.1±955
FPSO	3.7e+04±9.79e+03	21.3±0.122	1.04e+03±197	4.2e+03±2.47e+03	768±161	1.04e+03±197	141±77.6	21.3±0.14	1.95e+08±2.29e+08
TRIBES-D	3.55e+04±2.91e+03	21.3±0.0235	469±88.1	**1.29e-13**±2.51e-14	485±124	469±88.1	124±9.66	21.3±0.023	3.04e+03±4.37e+03
G-CMA-ES	2.96e+04±1.09e-11	21.3±1.78e-14	304±5.68e-14	1.71e-13±0	908±4.55e-13	304±5.68e-14	**0**±0	21.3±1.78e-14	3.99±0
f	f_{37}	f_{38}	f_{39}	f_{40}	f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
SLPSO	9.94±38.8	2.78e+05±2.56e+05	5.17e-13±3.44e-13	**340**±576	517±1.77e+03	1.91e+03±373	2.01e+03±1.49e+03	2e+03±1.48e+03	1.84e+03±228
APSO	2.09e+05±1.76e+06	5.62e+05±2.53e+06	3.36e+08±6.84e+09	703±2.49e+03	1.53e+03±3.52e+03	2.11e+03±489	2.21e+03±1.67e+03	2.25e+03±1.69e+03	2.32e+03±296
CLPSO	1.28e+05±5.16e+04	2.13e+05±8.38e+04	0.00279±0.00402	408±766	488±1.28e+03	1.82e+03±214	2.14e+03±1.57e+03	2.15e+03±1.58e+03	2.25e+03±74.2
CPSOH	3.68e+04±1.05e+05	1.47e+05±2.58e+05	1.54e+06±3.44e+07	951±508	1.48e+03±1.19e+03	1.51e+03±1.54e+03	2.23e+03±234	2.24e+03±187	2.28e+03±322
FIPS	1.40e+05±8.22e+04	1.95e+05±9.2e+04	237±1e+03	460±534	501±511	1.17e+03±81.4	1.78e+03±2.45e+03	1.7e+03±2.41e+03	575±141
SPSO	6.44e+03±1.05e+04	1.46e+05±1.65e+05	3.58e-13±7.84e-13	557±2.02e+03	457±1.29e+03	**977**±1.15e+03	1.61e+03±3.64e+03	1.85e+03±1.63e+03	997±3.5e+03
JADE	94.5±1.73e+03	8.68e+04±7.85e+04	**1.48e-13**±3.07e-13	401±847	497±1.77e+03	991±3.31e+03	1.94e+03±2.29e+03	1.89e+03±2.73e+03	945±4.26e+03
HRCGA	1.69e+04±7.8e+04	**8.39e+04**±8.09e+04	3.17e-05±5.84e-04	407±197	407±197	1.12e+03±222	**882**±3.66e+03	**836**±3.58e+03	**401**±26.7
FPSO	1.71e+05±7.17e+04	2.16e+05±1.04e+05	1.44e+07±9.51e+07	988±405	1.17e+03±588	1.56e+03±1.03e+03	2.07e+03±1.55e+03	2.08e+03±1.58e+03	2.15e+03±734
TRIBES-D	6.07e+03±1.26e+03	9.35e+04±1.28e+04	1.34e+07±2.51e+06	773±502	789±609	980±612	900±0	900±0	900±0
G-CMA-ES	**3.41e-13**±0	1.67e+05±1.26e+05	1.71e-13±0	400±1.71e-13	**400**±1.14e-13	2.01e+03±14.6	2e+03±6.82e-13	1.99e+03±2.27e-13	2.08e+03±1.82e-12

TABLE XIII
STATISTICAL RESULTS OF THREE DIFFERENT ASPECTS REGARDING THE NUMBER OF BEST MEAN VALUES ACHIEVED (#BM), THE NUMBER OF SOLVED (#S), PARTIALLY SOLVED (#PS), NEVER SOLVED PROBLEMS (#NS), AND t-TEST RESULTS OVER THE 45 PROBLEMS IN 10, 30, 50 AND 100 DIMENSIONS, WHERE "#+", "#-", AND "#∼" REPRESENT THAT THE PERFORMANCE OF SLPSO IS SIGNIFICANTLY BETTER THAN, SIGNIFICANTLY WORSE THAN, AND STATISTICALLY EQUIVALENT TO THE PERFORMANCE OF ITS RIVAL IN TERMS OF THE t-TEST RESULTS, RESPECTIVELY

	Dim	SLPSO	APSO	CLPSO	CPSOH	FIPS	SPSO	JADE	HRCGA	FPSO	TRIBES-D	APrMF	G-CMA-ES
#BM	10,30,50,100	11,11,12,14	1,0,0,0	5,0,0,0	0,0,0,0	0,0,0,0	0,0,0,1	0,0,0,3	4,4,5,8	0,0,0,0	12,17,14,16	4,8,1,-	8,11,13,6
#S,#PS,#NS	10	22,9,14	17,13,15	11,17,17	14,15,16	9,17,19	11,11,23	10,20,15	14,12,19	10,15,20	19,6,20	14,6,25	18,1,26
	30	19,6,20	7,13,25	10,9,26	6,8,31	6,8,31	7,9,29	9,15,21	10,7,28	0,2,43	18,3,24	7,7,31	17,1,27
	50	20,5,20	5,13,27	11,10,24	6,9,30	8,6,31	7,6,32	9,15,21	11,4,30	0,0,45	18,3,24	3,5,33	17,0,28
	100	18,5,22	0,14,31	7,0,38	5,3,37	1,0,44	7,5,33	8,11,26	8,6,31	0,1,44	18,2,25	-	18,0,27
#+,#-,#∼	10	-	22,2,21	12,10,23	29,0,16	22,6,17	26,3,16	15,7,23	12,12,21	29,4,12	17,10,18	23,7,15	28,12,5
	30	-	21,8,16	30,5,10	28,5,12	30,7,8	25,9,11	17,17,11	19,9,17	36,3,6	19,16,10	27,9,9	24,17,4
	50	-	27,3,15	31,9,5	33,6,6	29,9,7	24,13,8	18,17,10	22,16,7	35,5,5	16,19,10	31,6,8	17,19,9
	100	-	33,1,11	33,7,5	35,7,3	37,6,2	24,16,5	18,18,9	22,14,9	38,3,4	19,21,5	-	19,20,6

problems. Compared with other algorithms except TRIBES-D, SLPSO is the only algorithm that successfully found the global optimum of the Rastrigin (f_2) and non-continuous Rastrigin (f_3) functions over all the four dimensional cases. Especially for the Schwefel function (f_6), SLPSO is the only algorithm that is able to find the near global optima across the four dimensional cases among the 12 algorithms. SLPSO also achieves the best results on function f_{12} among all the algorithms. Functions f_1, f_7, and f_9 are successfully solved by all algorithms in all the tested dimensions under the given accuracy level, except SPSO on functions f_7 and APSO and FPSO on function f_9 in 50 and 100 dimensions. For function

f_8, G-CMA-ES achieves the best results that are much better than the other algorithms.

For the modified problems, the performance comparison of each algorithm on the original problems and corresponding modified problems is not shown due to the space limitation. We only summarize the comparison as follows: 1) different modifications on a specific problem bring in different difficulties; 2) the same modification on different problems also brings in different difficulties; 3) the same modification on a specific problem brings in different difficulties for different algorithms.

From the experimental results, it can be seen that modifications do raise the challenge to most algorithms and make the problems harder to solve than the original problems. One interesting thing is that, among the 12 algorithms, SLPSO is the only algorithm that is not sensitive to the modification of shifting the global optimum to a random location on the four test functions. Although TRIBES-D successfully finds the global optima for most traditional problems whose global optima have the same parameter values in all dimensions (e.g., the Rosenbrock function f_8 has a global optimum of [1,1,...,1]), it fails to find the global optima of the corresponding modified problems. Taking function f_2 in 30 dimensions as an example, TRIBES-D successfully finds the global optima of function f_2, while for the shifted f_2 (i.e., f_{17}), it achieves the second worst results among all the tested algorithms. The similar observation can be made with APrMF on function f_1 and f_{18}.

For the composition functions f_{13} and f_{14}, G-CMA-ES obtains the best results in most dimensional cases among the 12 algorithms. HRCGA also achieves good performances, which are slightly worse than the performance of G-CMA-ES. Except G-CMA-ES and HRCGA, SLPSO achieves much better results than the other 12 algorithms on most dimensional cases. For the other composition functions (f_{40}-f_{45}), the performance of SLPSO is not the best among the 12 algorithms. But, it does not mean that SLPSO is not suitable to solve this kind of problems. Comparing the results of SLPSO with the optimal configurations and with the default configurations in Table IX, it can be seen that the performances of SLPSO with the optimal configurations are much better than that of SLPSO with the default configurations. Generally speaking, among the 12 algorithms, HRCGA, APrMF, and SPSO have relatively better performances on the composition problems.

Based on the results in the first row of Table XIII, it can be seen that TRIBES-D, SLPSO, and G-CMA-ES obviously dominate the other algorithms in terms of the number of the best mean results. Another interesting observation is that the performance of APSO and CLPSO dramatically decreases with the increasing of the number of dimensions, but the situation of SLPSO and JADE is opposite to APSO and CLPSO and their performance increases with the increasing of the number of dimensions. For example, the number of the best mean results obtained by SLPSO on problems in 10 dimensions is 11 and the figure increases to 14 on problems in 100 dimensions.

B. Comparison Regarding the Success Rate

According to the accuracy level given for each problem in this paper, we present the comparison of the statistical results of the 12 algorithms in terms of the success rate. The second row in Table XIII shows the number of problems that are solved (#S), partially solved (#PS), and never solved (#NS) by the 12 algorithms in 10, 30, 50, and 100 dimensions. A problem is called solved, partially solved, or never solved, if an algorithm reaches the corresponding given accuracy level over all 30 runs, over some of (but not all) 30 runs, or over none of 30 runs, respectively.

In the second row of Table XIII, the number of solved, partially solved, or never solved problems is calculated by the number of problems where the success rate is equal to 0, within (0, 1), or equal to 1, respectively, for each algorithm. It can be seen that SLPSO has the best performance among the 12 algorithms on problems in all the dimensional cases. TRIBES-D achieves slightly worse results than SLPSO. The number of problems solved by SLPSO is about 20 in the four dimensional cases, which is about twice as the number of problems solved by the fourth best algorithm (CLPSO). The number of problems solved by G-CMA-ES is slightly smaller than that of TRIBES-D, which makes it the third best algorithm.

Generally speaking, the difficulty of a problem will increase when the number of dimensions increases. So, an algorithm's performance will also decrease. From the results in the second row of Table XIII, we can clearly see this trend for some algorithms, e.g., APSO, CPSOH, FIPS, FPSO, and APrMF, where the number of never solved problems increases while the number of solved problems decreases when the number of dimensions increases. For algorithms SLPSO, CLPSO, SPSO, JADE, HRCGA, TRIBES-D, and G-CMA-ES, their performances remain at a certain level when the number of dimensions increases.

C. Comparison Regarding the t-Test Results

In order to investigate how much the performance of SLPSO is better or worse than the performance of other 11 algorithms at the statistical level on each problem, a two-tailed t-test operation was performed in this section. The statistical results are shown in the third row in Table XIII – the detailed results are not provided in this paper due to the space limit. Each result is shown as #+, #-, and #$\sim$, which mean that the performance of SLPSO is significantly better than, significantly worse than, and statistically equivalent to the performance of its rival in terms of the t-test results, respectively. For example, the t-test results between SLPSO and TRIBES-D on 50 dimensions are (16,19,10), which mean that SLPSO achieves significantly better results than, significantly worse results than, and statistical equivalent results to TRIBES-D on 16, 19, and 10 problems, respectively.

Compared with the other 11 algorithms, it can be seen that SLPSO outperforms the other 11 algorithms in terms of the t-test results except the TRIBES-D and G-CMA-ES algorithms in 50 and 100 dimensions. The number of problems where SLPSO achieves significantly better results than the other

TABLE XIV
ALGORITHMS' RANKING REGARDING THE NUMBER OF BEST MEAN VALUES (#BM), PROBLEMS SOLVED (#S), AND t-TEST RESULTS (t-TEST)

	#BM(DIM)				#S(DIM)				t-Test(DIM)				Overall
	10	30	50	100	10	30	50	100	10	30	50	100	
SLPSO	2	2	3	2	1	1	1	1	1(111)	1(134)	1(144)	1(162)	1(1.42)
APSO	6	5	6	7	4	6	9	6	6(66)	7(85)	10(53)	11(21)	9(6.92)
CLPSO	4	5	6	7	6	4	4	3	2(105)	6(86)	6(108)	7(104)	6(5.0)
CPSOH	7	5	6	7	5	7	8	4	10(38)	11(62)	9(63)	9(68)	11(7.33)
FIPS	7	5	6	7	8	7	6	5	8(55)	9(65)	8(74)	8(71)	10(7.0)
SPSO	7	5	6	6	6	6	7	3	9(42)	8(82)	7(102)	6(134)	8(6.42)
JADE	7	5	6	5	7	5	5	2	5(73)	2(117)	2(135)	2(157)	5(4.42)
HRCGA	5	4	4	4	5	4	4	2	3(88)	5(96)	4(124)	5(139)	4(4.33)
FPSO	7	5	6	7	7	8	11	6	11(32)	12(31)	12(34)	10(44)	12(8.5)
TRIBES-D	1	1	1	1	2	2	2	1	4(84)	3(113)	3(130)	3(150)	2(2.0)
APrMF	5	3	5	-	5	6	10	-	7(64)	10(64)	11(46)	-	7(5.64)
G-CMA-ES	3	2	2	3	3	3	3	1	6(66)	4(111)	5(114)	4(142)	3(3.25)

algorithms is much larger than the number of problems where SLPSO performs significantly worse than the other algorithms.

D. Algorithms' Ranking

In order to have an overall view of the performances of the 12 algorithms, we rank the 12 algorithms regarding three different aspects: a) the number of problems where the best result is achieved; b) the number of solved problems; c) the number of problems where the performance of an algorithm is significantly better than its peer algorithms according to the t-test values. For the second criterion, if two algorithms have the same number of solved problems, then check the number of partially solved problems. If they still have the same number, it means they have the same ranking.

Table XIV shows the ranking of each algorithm in all the four dimensional cases, where the overall ranking is calculated by sorting the average of the sum of all the rankings obtained of each algorithm. For the fourth column in Table XIV, the value attached to each ranking value is the number of problems where the performance is significantly better than its peer algorithms according to the t-test values. It should be noted that we just use these methods to show the overall performance of the involved algorithms even though these ranking criteria might not be fair enough to show the performance of each algorithm.

According to the algorithms' ranking, it can be seen that SLPSO outperforms all the other algorithms in terms of different aspects except the second ranking with the best mean values in 10, 30, and 100 dimensions and the third ranking in 50 dimensions. Finally, the rankings of all the involved algorithms regarding the three different aspects are as follow: SLPSO, TRIBES-D, G-CMA-ES, HRCGA, JADE, CLPSO, APrMF, SPSO, APSO, FIPS, CPSOH, and FPSO.

E. Comparison on Two Real-World Problems

In order to test the effectiveness of SLPSO on real-world applications, we chose two problems from real life: design of a gear train [35] and parameter estimation for frequency-modulated (FM) sound waves. The first problem, which was introduced in [35], is to optimize the gear ratio for a compound gear train that contains three gears. It is to be designed that the gear ratio is as close as possible to 1/6.931. For each gear, the number of teeth must be between 12 and 60. The mathematical model of this problem can be described as follows:

$$f(x) = (1/6.931 - \frac{x_1 \cdot x_2}{x_3 \cdot x_4})^2 \quad (18)$$

where $x_i \in [12, 60], i = 1, 2, 3, 4$.

The second problem is to estimate the parameters of a FM synthesizer [10]. This problem is a highly complex multi-modal problem with strong epistasis. The parameter vector has six components: $X = [a_1, \omega_1, a_2, \omega_2, a_3, \omega_3]$, which is given in the following equation:

$$y(t) = a_1 \cdot sin(\omega_1 \cdot t \cdot \theta + a_2 \cdot sin(\omega_2 \cdot t \cdot \theta + a_3 \cdot sin(\omega_3 \cdot t \cdot \theta))) \quad (19)$$

and the expression of the target sound waves is given by:

$$y_0(t) = sin(5 \cdot t \cdot \theta + a1.5 \cdot sin(4.8 \cdot t \cdot \theta + 2 \cdot sin(4.9 \cdot t \cdot \theta))) \quad (20)$$

where $\theta = 2\pi/100$ and the parameters are constrained in $[-6.4, 6.35]$. The fitness function is defined as follows:

$$f(\vec{x}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \quad (21)$$

All the algorithms involved in previous experiments were tested on these two problems. For each algorithm, the parameter settings used were the same as in previous experiments and the swarm size was set to 10 for all algorithms except for the TRIBES-D, G-CMA-ES, and HRCGA algorithms. The total number of fitness evaluations was set to 20000 and 30000 for the first and second problems, respectively. The statistical results over 30 runs are shown in Table XV, where "Min", "Max", "Mean", and "Std" are the best, worst, mean, and standard deviation values, respectively.

From Table XV, it can be seen that the first problem has been solved easily by all the algorithms. However, for the second problem, none of the 12 algorithms finds the global optimum for all the 30 runs. By observing the Min values with the second problem, SLPSO, APSO, FIPS, SPSO, JADE, HRCGA, and FPSO have found the global optimum at least once in 30 runs. However, TRIBES-D, CLPSO, CPSO-H, APrMF, and G-CMA-ES have never found the global optimum. Among the seven algorithms that are able to find the global optimum at least once, SLPSO obtains the smallest mean error.

TABLE XV
COMPARISON ON TWO REAL-WORLD PROBLEMS

	Gear ratio				Estimation error			
	Min	Max	Mean	Std	Min	Max	Mean	Std
SLPSO	2.70e-12	6.19e-09	2.22e-09	9.83e-09	0	13.79	4.18	26.99
APSO	2.70e-12	1.31e-08	1.59e-09	1.44e-08	0	34.22	11.33	41.13
CLPSO	2.70e-12	1.36e-09	1.99e-10	2.22e-09	0.007	14.08	3.82	23.53
CPSOH	1.54e-10	2.02e-06	2.80e-07	2.33e-06	3.45	42.52	27.08	60.61
FIPS	8.88e-10	8.9e-07	3.27e-08	8.77e-07	0	15.11	5.93	25.75
SPSO	2.70e-12	2.56e-07	1.39e-08	2.57e-07	0	18.27	9.88	33.85
JADE	2.70e-12	1.36e-09	2.10e-10	2.259e-09	0	13.92	7.55	26.18
HRCGA	2.70e-12	1.181e-09	1.53e-10	1.88e-09	0	17.59	8.41	32.54
FPSO	2.057e-09	0.00017	1.57e-05	0.00018	0	15.82	5.22	28.31
TRIBES-D	9.64e-12	7.17e-05	3.62e-06	1.26e-05	2.22	22.24	14.68	4.57
APrMF	2.70e-12	2.36e-09	1.01e-09	6.84e-10	0.063	0.94	0.54	0.22
G-CMA-ES	2.701e-12	7.32e-001	2.44e-02	1.31e-01	3.326	55.09	38.75	16.77

VII. Conclusions

This paper investigates a self-learning PSO (SLPSO) algorithm that can enable a particle to adaptively adjust its search behavior during the search process for global optimization problems. In SLPSO, each particle has four learning sources produced by four operators, which have different properties to guide particles to converge to the current global best position, exploit a local optimum, explore new promising areas, and jump out of a local optimum, respectively. An adaptive selection mechanism is introduced to enable particles to automatically choose the appropriate learning objective at the appropriate moment during the search process.

Experiments on some challenging test problems were carried out in this paper. From the comparison results on all the test problems, several conclusions can be drawn for the SLPSO algorithm. First, the adaptive learning mechanism works at the individual level where particles are independent to adapt with their local fitness landscapes in different sub-regions. Each particle can choose an appropriate strategy at an appropriate moment according to the property of its local search space for achieving the best performance of SLPSO. Second, SLPSO significantly enhances the performance of PSO in terms of the performance metrics used in this paper and SLPSO performs much better than other peer algorithms. Third, SLPSO is also an effective optimization tool for the two real-world applications tested in this paper.

We can not expect SLPSO to solve every global optimization problem. In fact, it is impossible. However, the adaptive learning framework attempts to provide a special approach where particles have more intelligence to decide their own step direction based on the knowledge learnt from the local fitness landscape. It is helpful for solving different types of problems, especially problems that have very complex fitness landscapes.

Acknowledgment

The authors would like to thank Prof. X. Yao, Prof. Y. Jin and Dr. P. Rohlfshagen for their thoughtful suggestions and constructive comments.

References

[1] P. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *7th Conf. Evol. Programming*, 1998, pp. 601–610.

[2] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *2005 Congr. Evol. Comput.*, vol. 2, 2005, pp. 1769–1776.

[3] T. M. Blackwell and P. Bentley, "Don't push me! collision-avoiding swarms," in *2002 Congr. Evol. Comput.*, vol. 2, 2002, pp. 1691–1696.

[4] T. M. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, 2006.

[5] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *IEEE Swarm Intel. Symp.*, 2007, pp. 120–127.

[6] R. Brits, A. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *4th Asia-Pacific Conf. Simulated Evolution and Learning*, vol. 2, 2002, pp. 692–696.

[7] Y. Chen, W. Peng, and M. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Trans. Syst.,Man, Cybern. B: Cybern.*, vol. 37, no. 6, pp. 1460–1470, 2007.

[8] Y. Cooren, M. Clerc, and P. Siarry, "Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm," *Swarm Intell.*, vol. 3, pp. 149–178, 2009.

[9] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 913–920.

[10] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problem," Dept. of Electronics and Telecommunication Engg., Jadavpur University, Kolkata, India, Tech. Rep., 2011.

[11] M. A. M. de Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's pso: A composite particle swarm optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1120–1132, 2009.

[12] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.,*, vol. 3, no. 2, pp. 124–141, 1999.

[13] C. Garcia-Martinez, M. Lozano, F. Herrera, D. Molina, and A. M. Sanchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *Europ. J. Oper. Res.*, vol. 185, pp. 1088–1113, 2008.

[14] D. E. Goldberg, "Probability matching, the magnitude of reinforcement,and classifier system bidding," *J. Mach. Learn.*, vol. 5, no. 4, pp. 407–425, 1990.

[15] S. Hsieh, T. Sun, C. Liu, and S. Tsai, "Efficient population utilization strategy for particle swarm optimizer," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 39, no. 2, pp. 444–456, 2009.

[16] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 35, no. 6, pp. 1272–1282, 2005.

[17] G. Kendall, E. Soubeiga, and P. Cowling, "Choice function and random hyper heuristics," in *Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, SEAL*. Springer, 2002, pp. 667–671.

[18] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *1999 Congr.Evol. Comput.*, 1999, pp. 1931–1938.

[19] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[20] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *2002 Congr. Evol. Comput.*, 2002, pp. 1671–1676.

[21] C. Li and S. Yang, "An adaptive learning particle swarm optimizer for function optimization," in *2009 Congr. Evol. Comput.*, 2009, pp. 381–388.

[22] C. Li, S. Yang, and I. A. Korejo, "An adaptive mutation operator for particle swarm optimization," in *2008 UK Workshop Comput. Intell.*, 2008, pp. 165–170.

[23] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *2004 Genetic Evol. Comput. Conf.*, 2004, pp. 105–116.

[24] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[25] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baska, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, 2006.

[26] J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *2005 Symp. Swarm Intell.*, 2005, pp. 68–75.

[27] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag, "Extreme compass and dynamic multi-armed bandits for adaptive operator selection," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, may 2009, pp. 365 –372.

[28] C. Maurice, "Standard PSO 2007 (SPSO-07)," http://www.particleswarm.info/Programs.html, 2007.

[29] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, Maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 204–210, 2004.

[30] Q. H. Nguyen, Y. S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, 2009, source codes available at: http://code.google.com/p/memetic-algorithm/downloads/list.

[31] T. Niknam and E. A. Farsani, "A hybrid self-adaptive particle swarm optimization and modified shuffled frog leaping algorithm for distribution feeder reconfiguration," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 8, pp. 1340 – 1349, 2010.

[32] R. Poli, C. D. Chio, and W. B. Langdon, "Exploring extended particle swarms: a genetic programming approach," in *2005 Conf. on Genetic and Evol. Comput.*, 2005, pp. 33–57.

[33] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp. 33–58, 2007.

[34] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, 2004.

[35] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design," in *the ASME Design Technology Conf.*, 1988, pp. 95–105.

[36] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *IEEE Int. Conf. Evol. Comput.*, 1998, pp. 69–73.

[37] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization," in *2001 Congr. Evol. Comput.*, vol. 1, 2001, pp. 101–106.

[38] J. E. Smith, "Credit assignment in adaptive memetic algorithms," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 1412–1419.

[39] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 1, pp. 81–87, 1997.

[40] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *1999 Congr. Evol. Comput.*, 1999, pp. 1958–1962.

[41] P. N. Suganthan, N. Hansen, J. J. Liang, Y.-P. C. K. Deb, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2005.

[42] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO'05. New York, NY, USA: ACM, 2005, pp. 1539–1546.

[43] D. Thierens, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, F. Lobo, C. Lima, and Z. Michalewicz, Eds., vol. 54. Springer Berlin / Heidelberg, 2007, pp. 77–90.

[44] F. van den Bergh and A. P. Engelbrech, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 225–239, 2004.

[45] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, Pretoria, South Africa, South Africa, 2002.

[46] C. Wei, Z. He, Y. Zhang, and W. Pei, "Swarm directions embedded in fast evolutionary programming," in *2002 Congr. Evol. Comput.*, vol. 2, 2002, pp. 1278–1283.

[47] T. White, "Swarm intelligence," http://www.sce.carleton.ca/netmanage/tony/swarm.html, 1997.

[48] Z. Wu and J. Zhou, "A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment," in *2007 Conf. Comput. Intell. and Security*, 2007, pp. 133–136.

[49] X. Xie, W. Zhang, and Z. Yang, "Dissipative particle swarm optimization," in *2002 Congr. Evol. Comput.*, vol. 2, 2002, pp. 1456–1461.

[50] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, 2010.

[51] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[52] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 39, pp. 1362–1381, 2009.

[53] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

Changhe Li received the B.Sc. and M.Sc. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from the University of Leicester, U.K. in July 2011. He is currently a lecturer with the School of Computer Science, China University of Geosciences, Wuhan, China. His research interests are evolutionary algorithms, particle swarm optimization, and dynamic optimization.

Shengxiang Yang received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China in 1993, 1996, and 1999, respectively. He is currently a Senior Lecturer with the Department of Information Systems and Computing, Brunel University, U.K. He has over 130 publications. He has given invited keynote speeches in several international conferences and co-organised several symposiums, workshops and special sessions in conferences. He serves as the area editor, associate editor or editorial board member for four international journals. He has co-edited several books and conference proceedings and co-guest-edited several journal special issues. His major research interests include evolutionary and genetic algorithms, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling and real-world applications. He is the chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, Evolutionary Computation Technical Committee, IEEE Computational Intelligence Society and the founding chair of the Task Force on Intelligent Network Systems, Intelligent Systems Applications Technical Committee, IEEE Computational Intelligence Society.

Trung Thanh Nguyen received his BSc in Computer Science from Vietnam National University, Hanoi in 2000, and his MPhil and PhD in Computer Science from the University of Birmingham in 2007 and 2011, respectively. From 2000 to 2005 he was a researcher in the Research Institute of Post and Telecoms in Vietnam. In 2011 he was a research fellow in the University of Birmingham. He is currently a research fellow in the Liverpool John Moores University, UK. His current research interests are global (dynamic and static) optimisation using metaheuristics and bio-inspired methods, and applications of metaheuristics and operation research to real-world problems, especially problems in port/maritime evironments.