Triple Archives Particle Swarm Optimization

Xuewen Xia, *Member, IEEE*, Ling Gui, Fei Yu, Hongrun Wu, Bo Wei, Ying-Long Zhang, and Zhi-Hui Zhan, *Senior Member, IEEE*

***Abstract*—There are two common challenges in particle swarm optimization (PSO) research, that is, selecting proper exemplars and designing an efficient learning model for a particle. In this article, we propose a triple archives PSO (TAPSO), in which particles in three archives are used to deal with the above two challenges. First, particles who have better fitness (i.e., elites) are recorded in one archive while other particles who offer faster progress, called profiteers in this article, are saved in another archive. Second, when breeding each dimension of a potential exemplar for a particle, we choose a pair of elite and profiteer from corresponding archives as two parents to generate the dimension value by ordinary genetic operators. Third, each particle carries out a specific learning model according to the fitness of its potential exemplars. Furthermore, there is no acceleration coefficient in TAPSO aiming to simplify the learning models. Finally, if an exemplar has excellent performance, it will be regarded as an outstanding exemplar and saved in the third archive, which can be reused by inferior particles aiming to enhance the exploitation and to save computing resources. The experimental results and comparisons between TAPSO and other eight PSOs on 30 benchmark functions and four real applications suggest that TAPSO attains very promising performance in different types of functions, contributing to both higher solution accuracy and faster convergence speed. Furthermore, the effectiveness and efficiency of these new proposed strategies are discussed based on extensive experiments.**

***Index Terms*—Elite particles, global optimization, particle swarm optimization (PSO), profiteer particles, triple archives.**

Manuscript received April 27, 2019; revised July 30, 2019; accepted September 16, 2019. Date of publication October 11, 2019; date of current version December 3, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61663009, Grant 61806204, Grant 61762036, Grant 61822602, and Grant 61772207, in part by the Foundation of Fujian Province Great Teaching Reform under Grant FBJG20180015, and in part by the National Natural Science Foundation of Jiangxi Province under Grant 20171BAB202012. This article was recommended by Associate Editor Y. Tan. *(Corresponding author: Zhi-Hui Zhan.)*

X. Xia, L. Gui, F. Yu, H. Wu, and Y.-L. Zhang are with the College of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China (e-mail: xwxia@whu.edu.cn; 983189239@qq.com; phoenix@mnnu.edu.cn; dr.hongrunwu@gmail.com; zhang_yinglong@126.com).

B. Wei is with the School of Software, East China Jiaotong University, Nanchang 330013, China (e-mail: weibo@whu.edu.cn).

Z.-H. Zhan is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2943928

I. Introduction

THE PARTICLE swarm optimization (PSO) algorithm, inspired by the swarm behaviors of birds flocking, was proposed by Kennedy and Eberhart in 1995 [1], [2]. In canonical PSO, each particle represented by a point in the Cartesian coordinate system chooses the global best-so-far solution as well as its personal best experience as learning exemplars. Although a single particle has very low intelligence, the collective behavior of different particles can display a powerful ability in solving complicated problems [3]–[5].

The ability to deal with the complex problems that emerged from the collective behaviors depends on the particles' learning models, which decide information interactive models between different particles. Generally, a high efficient learning model in PSO hinges on proper exemplars and learning models of each particle. In canonical PSO [1], [2], a learner particle selects the global or the local best particle, measured by fitness, as its exemplar to perform the learning process. Generally, when optimizing simple unimodal functions, the fitness-based selection of exemplars is a very promising and efficient option due to the outstanding exemplars, which can be regarded as elites, and can help the learner particle find out the global optima with high convergence speed. However, the fitness-based selection method may cause the population to be easily trapped in local optima when optimizing complicated multimodal functions.

Thus, different characteristics are adopted as criteria [6]–[9] to choose exemplars for a learner particle. For instance, in [7], a particle uses the ratio of the relative fitness and the distance of other particles, rather than the fitness, to determine its exemplar. The novel selection of exemplars offers less susceptibility to premature convergence. To overcome the "two steps forward, one step back" phenomenon caused by the fitness-based selections of exemplars, Zhan *et al.* [10] used an orthogonal learning strategy to generate exemplars. Extensive experimental results verify that these PSO variants based on the nonfitness selection of exemplars alleviate the contradiction between exploration and exploitation to some extent.

Generally, from the perspective of the fitness landscape, areas with greater steep surface curves are common around local optima [11]. In other words, when flying around local optima, a particle's fitness may be dramatically changed in two consecutive generations. In this article, the particle is called a profiteer. Considering that the global optimum must be a local optimum, if there is thus a profiteer near the global optimum, taking advantage of helpful information implied in the

profiteer is beneficial for optimizing the complicated functions. In addition, since the fitness-based selection of exemplars has favorable performance on the unimodal functions, rationally utilizing both elites and profiteers is a promising way to enhance the comprehensive performance of PSOs.

Moreover, considering that all particles using the same learning model may cause a lack of intelligence in populations to cope with different complex situations, some researchers expand the mono-learning-model to the multi-learning model [12], [13]. Many studies verify that assigning multiple learning models for different particles based on the characteristics of the current fitness landscape is beneficial for difficult functions.

Based on the above considerations, this article proposes a new PSO variant, called triple archives PSO (TAPSO). In TAPSO, two archives are used to save elites and profiteers, respectively. In each generation, two particles, respectively, selected from the two archives are used to breed an exemplar for a learner particle. Based on the performance of the exemplar, the learner particle adopts an appropriate learning model. Furthermore, to save computing resources, those outstanding exemplars are saved in the third archive, which can be reused by inferior particles. The main characteristics of TAPSO can be summarized as follows.

1) Instead of using fitness as a single criterion, the improvement rate of fitness is considered as an additional criterion when choosing learning exemplars for a particle.
2) Three external archives are, respectively, used to save elites, profiteers, and outstanding exemplars generated by the elites and profiteers through crossover and mutation operators. Due to the crossover operator, the new generated exemplar contains both promising information of elites and profiteers while the mutation operator injects diversity information into the exemplar.
3) In each generation, a particle adopts its suitable learning model according to the performance of the generated exemplar. In this case, the particle can conduct different search behaviors in different generations aiming to deal with various fitness landscapes.
4) The generated outstanding exemplars saved in the third archive can be reused by those inferior particles aiming to enhance the exploitation and save the computing resources.

The remainder of this article is organized as follows. Section II presents a framework of the canonical PSO and reviews some PSO variants. The details of TAPSO are described in Section III. The experimental results and corresponding discussions are detailed in Section IV. Finally, the conclusions are presented in Section V.

II. Related Works

A. Canonical PSO

In PSO, each particle i at generation t is associated with two vectors, that is, a position vector $\mathbf{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,D}^t]$ and a velocity vector $\mathbf{V}_i^t = [v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,D}^t]$, where D represents the dimension of a problem under study. The vector $\mathbf{X}_i^t$ is regarded as a candidate solution while the vector $\mathbf{V}_i^t$ is treated as a search direction and step size of particle i at generation t. During the evolutionary process, each particle adjusts its flight trajectory based on two vectors, called personal historical best position $\mathbf{PB}_i^t = [pb_{i,1}^t, pb_{i,2}^t, \ldots, pb_{i,D}^t]$ and its neighbors' best-so-far position $\mathbf{NB}_i^t = [nb_{i,1}^t, nb_{i,2}^t, \ldots, nb_{i,D}^t]$, respectively. The update rules of $\mathbf{V}_i^t$ and $\mathbf{X}_i^t$ are defined as (1) and (2), respectively

$$v_{i,j}^t = w \cdot v_{i,j}^{t-1} + c_1 \cdot r_{1,j} \cdot \left(pb_{i,j}^t - x_{i,j}^{t-1}\right) + c_2 \cdot r_{2,j} \cdot \left(nb_{i,j}^t - x_{i,j}^{t-1}\right) \quad (1)$$

$$x_{i,j}^t = x_{i,j}^{t-1} + v_{i,j}^t \quad (2)$$

where w represents an inertia weight determining how much the previous velocity is preserved; c_1 and c_2 are two acceleration coefficients deciding relative learning weights for $\mathbf{PB}_i^t$ and $\mathbf{NB}_i^t$, respectively; $r_{1,j}$ and $r_{2,j}$ are two random numbers uniformly distributed in the interval [0, 1]; and $x_{i,j}^t$ and $v_{i,j}^t$ represent the jth dimension values of $\mathbf{X}_i^t$ and $\mathbf{V}_i^t$, respectively. Note that when a particle regards all other particles as its neighbors, $\mathbf{NB}_i^t$ is the current global best position $\mathbf{GB}$.

B. Studies of PSO

In the PSO community, designing a more efficient velocity update rule has captured many researchers' attention, and various PSO variants have been proposed during the last decades. Depending on the objectives to be dealt with, most studies can generally be classified into three categories, that is, parameter adjustment, learning strategy adjustment, and hybridization strategy, which are briefly reviewed hereinafter.

1) Parameters Adjustment: It is widely accepted that a smaller w facilitates the exploitation while a larger one is beneficial for the exploration. The most ubiquitous update rule of w linearly decreases from 0.9 to 0.4 over the optimization process which is still applied in many PSOs now [14]. Motivated by the iteration-based w, Ratnaweera *et al.* [15] further advocated a hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC). However, considering that the search process of PSO is nonlinear and complicated, various nonlinear-varying strategies are proposed to adjust the parameters [16]–[19] aiming to give particles diverse search behaviors.

Although the aforementioned strategies improve PSO's performance in various degrees, they may run into the risk of inappropriate tuning parameters because information of different evolutionary states is not appropriately utilized in the iteration-based strategies.

To lay out a more satisfactory adjustment for the parameters, many scholars proposed different adjustments by taking advantage of historical information of the entire population [9], [20], [21]. For instance, in adaptive PSO (APSO) [9], tuning w, c_1, and c_2 no longer relies on iteration numbers. Instead, an evolutionary state estimation, the evaluation of which is based on the distribution and the fitness of particles, is selected as a criterion to adjust the parameters. In [21], a new adaptive adjustment of w based on the Bayesian techniques is proposed to enhance the exploitation capability. A common feature in these improvements is that particles regulate their

own parameters according to their fitness [22]–[24], velocity [25], or population diversity [26], [27]. Extensive experiments demonstrate that the adaptive strategies can achieve a proper tradeoff between the exploration and exploitation abilities.

2) Learning Strategies Adjustment: In the PSO community, extracting information based on the global version topological structure PSO (GPSO) and the local version topological structure PSO (LPSO) are two basic learning strategies for particles [28]. Many studies [28]–[30] suggest that learning strategies based on sparse and dense neighbor topologies are beneficial for complicated multimodal problems and simple unimodal problems, respectively.

To overcome shortcomings caused by a single learning exemplar, many researchers adopt comprehensive information of the entire population as an exemplar for a particle by different weighting methods [6], [8], [10], [31]–[33]. In this sense, the comprehensive learning strategy [8], the orthogonal learning strategy [10], the interactive learning strategy [34], and the dimensional learning strategy [31] are remarkable works.

However, it is unrealistic to adopt an optimal static learning strategy with a fixed neighbor topology for a specific problem beforehand since many real applications are "black-box" problems and the optimization process is a dynamic course. Thus, quite a few dynamic learning strategies have been proposed in recent years. For instance, Peram *et al.* [7] proposed a fitness-distance-ration-based PSO (FDR-PSO) in which the Euclidian distance and fitness are deemed as criteria for a specific particle to adjust its learning exemplars. Other distance-based [35] and fitness-based [36] selections of exemplars also manifest very favorable performance. Furthermore, the multiswarm mechanism [37], [38] is also an efficient way to enhance the performance of PSO. The heterogeneous learning strategies based on the multiswarm mechanism enable the population to efficiently cope with different fitness landscapes and offer a better balance between exploration and exploitation [37]–[39].

To endow a population with more intelligence to solve different complex situations, Li *et al.* [13] proposed a self-learning PSO (SLPSO) in which particles are assigned four different roles according to distinct local fitness landscapes they belong to. Accordingly, the different roles representing four distinct learning strategies enable the particles to independently deal with various situations. The experiments demonstrate that the adaptive learning framework improves the local searching ability and overcomes the premature phenomena to some extent. Furthermore, learning strategies based on the pairwise competition mechanism [40] and exemplar pool strategy [41] also achieve a good balance between exploration and exploitation.

3) Hybridization of Auxiliary: Considering that different operators or optimization algorithms have their own characteristics, many researchers pour much attention on hybridizing them with reasonable integration strategies. For instance, genetic operators [42], [43] and various local searching strategies [41], [44], [45] are very popular auxiliaries for balancing the exploration and exploitation. Furthermore, Lévy flight, as a common random walk strategy, is another type of auxiliary for PSO [46], [47].

To take full advantage of some existing PSO variants, Lynn and Suganthan [48] proposed an ensemble PSO (EPSO), in which five PSO variants are hybridized together by an ensemble approach. Experiments verify that the adaptive mechanism employed to assign proper PSO variants to the population during the evolutionary process can organically integrate various virtues of the involved PSO variants. Apart from this, some studies manifest that hybridizing PSO with other evolutionary algorithms (EAs) also shows very promising performance [49]–[51]. No matter which cooperation mechanism is adopted in these type PSO variants, the main idea is using distinct search behaviors of involved algorithms to improve the exploration capability as well as sharing helpful information of the algorithms to enhance the exploitation capability.

Note that no matter which category a PSO variant belongs to, two issues must be considered, that is, selecting proper exemplars and designing an efficient learning model. Hence, TAPSO is detailed from the two perspectives in the next section.

III. TAPSO

A. Motivation of TAPSO

When using a heuristic algorithm to solve real applications, how to keep population diversity is a crucial problem [52], [53] since it determines the global searchability of the algorithms. In the canonical PSO, a particle only chooses one exemplar in its "social-learning" part. In this case, the particle cannot absorb much more helpful knowledge from various exemplars, which is harmful for population diversity. In fact, it is a common phenomenon in human society that people always have multiple exemplars during their social-learning process. Moreover, many studies indicate that children with multiple exemplars can offer better words learning ability compared to those children who only learn from one exemplar [54]–[56]. The simplest explanation is that an individual can extract more useful knowledge from multiple information providers than a single exemplar. Motivated by the studies, many scholars pour attention on generating a new promising exemplar for a particle based on multiple individuals' experience [6], [8], [42]. The experimental results testify the promising performance of these new generated exemplars which extract different knowledge from multiple individuals.

When selecting multiple exemplars for a particle, the majority of the above-mentioned studies chooses the fitness as a main criterion. Although this type of strategy can take advantage of much useful information implied in outstanding exemplars, overemphasizing the importance of the fitness value may increase the risk of premature convergence. Thus, it is urged for us to select helpful exemplars based on some new criteria.

There is a common phenomenon that we sometimes choose people who achieve the fastest progress as our exemplars since we believe that we can extract much useful experience from them though they do not display the most favorable

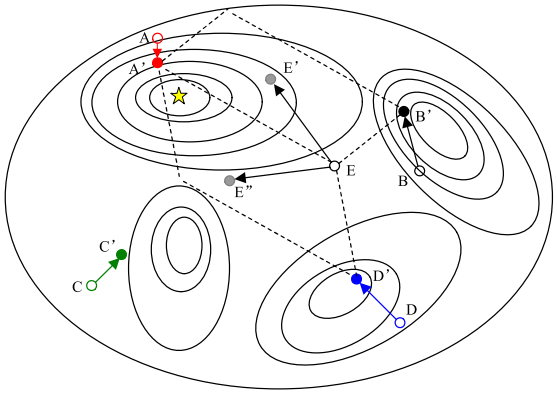


Fig. 1. Illustration of the learning process based on exemplars who have a favorable fitness and improvement rate.

performance. In fact, we can also discover the same characteristics of an optimization problem through the perspective of the fitness landscape. For instance, the fitness landscape of an area will not change significantly if no local optimum lies in the area. On the contrary, areas with very steep surface curves are common around local optima [11], [57]. Thus, when the fitness of a particle experiences a dramatic change in two consecutive generations, the particle, which is called a profiteer, may be located around a local optimum. Considering that the global optimum must be a local optimum, if there is a profiteer near the global optimum, reasonable utilization of the profiteer has potentially beneficial consequences.

In Fig. 1, although particles A', B', C', and D' are located around four different local optima, only A' is in the neighborhood of the global optimal solution (see the point ★). Furthermore, we can see that B' and D' have the best fitness in the current generation, while the particles A' and C' have the greatest and the lowest improvement rates after they update from A and C, respectively. If E only learns from the favorable particles with the highest fitness, that is, B' and D', E may fly away from the global optimal solution. On the contrary, if E selects A', who has the highest improvement rate, as an additional exemplar, it can fly toward the optimal solution. Specifically, when E chooses B' and A' (or D' and A') as its two learning exemplars, the particle will fly toward E' (or E'').

From Fig. 1, we can see that only particle A' that is near the global optimal solution can that particle E can find out the optimal solution. Thus, if we use an archive to save many particles that have great improvement rates, the probability that there is a particle located around the global optimal solution is very high. As a result, a learner particle can find out the optimal solution with high probability.

Inspired by the above discussions, TAPSO is proposed in this article. In TAPSO, the fitness and the improvement rate are two metrics to quantify a particle's characteristic. During the evolutionary process, particles who have a higher fitness or greater improvement rate are recorded in two archives, respectively. In each generation, two parents, respectively, selected from the two archives, are used to generate a versatile exemplar for a learner particle. Then, the learner particle conducts its own learning model to extract useful knowledge from the exemplar. Furthermore, to save computing resources, some outstanding generated exemplars are saved in the third archive, which can be reused by inferior particles.

To implement the aforementioned motivations, three main steps involved in TAPSO, that is; 1) breeding exemplars; 2) selecting learning model; and 3) reusing exemplars, are detailed as follows.

B. Breeding From Elite and Profiteer Archives

Without loss of generality, the considered objective f in this article is for minimization. In other words, the lower the value $f(\mathbf{X})$ is, the better performance $\mathbf{X}$ has. Thus, the improvement rate (Ir) of a particle i in TAPSO is defined as

$$Ir\left(\mathbf{X}_i^t\right) = \frac{f\left(\mathbf{X}_i^{t-1}\right) - f\left(\mathbf{X}_i^t\right)}{e^{\left|\mathbf{X}_i^{t-1} - \mathbf{X}_i^t\right|}} \tag{3}$$

where $f(\mathbf{X}_i^t)$ is the fitness of $\mathbf{X}_i^t$, and $|\mathbf{X}_i^{t-1} - \mathbf{X}_i^t|$ denotes the Euclidean distance between $\mathbf{X}_i^{t-1}$ and $\mathbf{X}_i^t$.

From (3), we see that a particle i with a higher Ir value, that is, a profiteer, represents that the particle yields a greater improvement with relatively small fly distance. Generally speaking, there may be a local optimum in an area if the fitness of a particle is changed significantly around the area.

Considering that elites and profiteers have their own advantages, two external archives are used to record them in each generation t, respectively. Specifically, archives A_e^t and A_p^t are used to save the better M elites and the better M profiteers, respectively. At each generation, A_e^t and A_p^t need to be updated according to f and Ir of the current particles, respectively. Specifically, when updating A_e^t at generation t, we first merge $\mathbf{X}_i^t$ with A_e^{t-1} as one set, and then use a greedy strategy to select those individuals with a better fitness value, that is, the f value, to save in the archive. Note that each element in A_e^t is unique. The update rule of A_p^t is similar to that of A_e^t except that the value of Ir is used as a metric when updating A_p^t.

To take advantage of both elites and profiteers when updating each dimension of a particle, two parents, respectively, selected from A_p^t and A_e^t, are used to breed a potential exemplar for the particle based on common genetic operators. Note that particles within A_e^t and A_p^t are sorted, in terms of f and Ir, respectively. Specifically, A_e^t and A_p^t are represented as (4) and (5), respectively, for minimization optimization problems

$$A_e^t = \left\{\mathbf{X}_{i_1}^t, \mathbf{X}_{i_2}^t, \ldots, \mathbf{X}_{i_M}^t | f\left(\mathbf{X}_{i_1}^t\right) \le f\left(\mathbf{X}_{i_2}^t\right) \le \cdots \le f\left(\mathbf{X}_{i_M}^t\right)\right\} \tag{4}$$

$$A_p^t = \left\{\mathbf{X}_{j_1}^t, \mathbf{X}_{j_2}^t, \ldots, \mathbf{X}_{j_M}^t | Ir\left(\mathbf{X}_{j_1}^t\right) \ge Ir\left(\mathbf{X}_{j_2}^t\right) \ge \cdots \ge Ir\left(\mathbf{X}_{j_M}^t\right)\right\}. \tag{5}$$

In this case, a particle with a smaller f or a greater Ir has a higher probability to be a parent. For the sorted A_e^t and A_p^t, the probability of the kth individual to be selected is calculated as follows:

$$p_k = \frac{M - k + 1}{\sum_{i=1}^{M} i} \tag{6}$$

where M is the size of the archives.

While a particle has good fitness and good improvement rate simultaneously, two copies of the particle need to be saved in

A_e^t and A_p^t at the same time. The reason why we save two copies of the promising particle is that we want to take full advantages of its two merits, that is, the better fitness and the greater improvement rate, when generating an exemplar. However, the shortcoming of the two copies saved in the two archives is that we may select the two copies simultaneously to generate an exemplar. In this case, the crossover operator [see (7)] has no effect, and the exemplar will be a mutation of the copy, that is, mutated by (8). However, even the particle appears in A_p^t and A_e^t at the same time, and the probability that the two copies to be simultaneously selected is low. From (6), we see that the probability that the two copies being simultaneously selected is the highest if the particle has the best fitness and the greatest improvement rate. In this case, if the archive size is 10, the probability is only 3.31%. While the particle does not have the best fitness and the greatest improvement rate at the same time, the probability will become very low. Thus, we regard saving two copies of a favorable particle to be beneficial for taking full advantage of its two characteristics implied in its better fitness and greater improvement rate.

In the following text, the process of breeding a potential exemplar is presented.

1) Crossover: In every generation, a potential exemplar $\mathbf{E}_i = [e_{i,1}, e_{i,1}, \ldots e_{i,D}]$ needs to be bred for a particle i. When generating the dth dimension of $\mathbf{E}_i$, called $e_{i,d}$, we use the roulette wheel selection based on (6) to select two parents $\mathbf{X}_{i_{p1}}$ and $\mathbf{X}_{i_{p2}}$ from A_e and A_p, respectively, and then conduct the crossover operator on the parents to breed $e_{i,d}$. The crossover operator is detailed as

$$e_{i,d} = \begin{cases} x_{i_{p2},d}, & \text{if } r_c < p_c \\ x_{i_{p1},d}, & \text{otherwise} \end{cases} \tag{7}$$

where r_c is a random number uniformly distributed in [0, 1]; p_c denotes a crossover rate; and $x_{i_{p1},d}$ and $x_{i_{p2},d}$ are the dth values of $\mathbf{X}_{i_{p1}}$ and $\mathbf{X}_{i_{p2}}$, respectively.

The value of p_c determines how many genes of a generated exemplar are selected from elites or profiteers and then bears on the exemplar's characteristics. We have investigated 11 different values of p_c and presented the results in Tables S-II and S-III in the supplementary material. The results indicate that setting p_c as 0.5 can achieve a balance between exploration and exploitation.

After the crossover operator, the exemplar $\mathbf{E}_i$ may have favorable characteristics since it integrates the outstanding merits of $\mathbf{X}_{i_{p1}}$ and $\mathbf{X}_{i_{p2}}$.

2) Mutation: After the crossover operator, a mutation operator is conducted on the bred offspring $\mathbf{E}_i$ with a mutation probability p_m. A greater p_m can inject more diversity into $\mathbf{E}_i$. On the contrary, a smaller p_m enables $\mathbf{E}_i$ to contain more useful information obtained from parents. The mutation operator applied in this article is the same as the classical mutation operator of GA. For each dimension d, for instance, if a random number $r_m \in [0, 1]$ is smaller than p_m, then $e_{i,d}$ is randomly reinitialized in the search space

$$e_{i,d} = \text{rand}(lb_d, ub_d), \text{if } r_m < p_m \tag{8}$$

Algorithm 1 Breed_Exemplar (A_e^t, A_p^t, p_m, p_c, p_k ($1 \le k \le M$))

```
For j = 1 to D Do
    Select X_{i_p1} from A_e^t based on probabilities p_k;
    Select X_{i_p2} from A_p^t based on probabilities p_k;
    Generate e_{i,j} according to Eq. (7);
    Carry out mutation for e_{i,j} according to Eq. (8);
End For
Output result: E_i = [e_{i,1}, e_{i,2}, ..., e_{i,D}].
```

where r_m is a random number uniformly distributed in [0, 1]; and lb_d and ub_d denote the lower and upper bounds of the dth dimension, respectively.

In many popular EAs, p_m in the interval [0.01, 0.05] is widely accepted. In this article, we have explored the characteristics of five different values of p_m in Tables S-IV and S-V in the supplementary material. The results indicate that setting p_m as 0.02 can achieve a balance between exploration and exploitation. Thus, $p_m = 0.02$ is adopted in all comparison experiments in this article.

By incorporating the aforementioned operators, the pseudocode of breeding an exemplar for a particle can be described as Algorithm 1.

C. Selecting Learning Model

After applying the crossover and mutation operators to generate a potential exemplar, particle i has three potential exemplars, that is, $\mathbf{PB}_i$, $\mathbf{GB}$, and $\mathbf{E}_i$. Thus, there are three ordering results of the three potential exemplars, in terms of the fitness. Accordingly, each particle i has three candidate learning models at each generation.

The first case is $f(\mathbf{E}_i) \le f(\mathbf{GB}) \le f(\mathbf{PB}_i)$. Considering that $\mathbf{E}_i$ contains favorable characteristics both of elites and profiteers, we regard that there should be a local (even a global) optimum solution around the excellent exemplar. Thus, it may be a profitable choice for the particle i to directly fly toward $\mathbf{E}_i$. This learning model can be called the *diffident* model since the particle i disregards its own historical experience, and completely depends on the best exemplar $\mathbf{E}_i$. With the *diffident* model, particle i flies toward $\mathbf{E}_i$ without other bias, which is beneficial for the exploitation. Moreover, after particle i updates its velocity and position, $\mathbf{GB}$ will be replaced by $\mathbf{E}_i$ which aims to save the outstanding solution.

The second case is $f(\mathbf{GB}) \le f(\mathbf{E}_i) \le f(\mathbf{PB}_i)$. Under this condition, $\mathbf{E}_i$ is better than $\mathbf{PB}_i$ though it is worse than $\mathbf{GB}$. In this case, assigning $\mathbf{E}_i$ as an exemplar for the particle i is a promising strategy not only for the favorable fitness of $\mathbf{E}_i$ extracted from elites but also for diversity information injected by the mutation operator [see (8)]. Moreover, to provide more direct guidance information for the particle i, $\mathbf{GB}$ is selected as another exemplar. Thus, this learning model based on $\mathbf{E}_i$ and $\mathbf{GB}$, called the *mild* model in this article, is a favor for offsetting contradictions between exploration and exploitation.

The last case is $f(\mathbf{GB}) \le f(\mathbf{PB}_i) \le f(\mathbf{E}_i)$. The result manifests that $\mathbf{E}_i$ does not contain enough helpful knowledge from elites and profiteers. Hence, particle i will only select $\mathbf{PB}_i$ as its exemplar to search for a promising region around itself. To simplify, this learning model is called the

confident model because particle i neglects the information of other particles, regardless of whether they are better than itself or not. Unlike the *diffident* model mentioned above, the *confident* learning model causes the particle i to ramble near $\mathbf{PB}_i$, which is beneficial for the exploration capability of the entire population.

According to the aforementioned analysis, the *diffident*, *mild*, and *confident* learning models detailed as follows are favorable for the exploitation, balanced search, and exploration abilities, respectively. Note that acceleration coefficients applied in the canonical PSO are removed from TAPSO, aiming to simplify the learning models.

1) *Diffident Model:* Learning from $\mathbf{E}_i$

$$v_{i,j}^t = w \cdot v_{i,j}^{t-1} + r_{1,j} \cdot \left(e_{i,j} - x_{i,j}^{t-1}\right). \tag{9}$$

2) *Mild Model:* Learning from $\mathbf{GB}$ and $\mathbf{E}_i$

$$v_{i,j}^t = w \cdot v_{i,j}^{t-1} + r_{1,j} \cdot \left(e_{i,j} - x_{i,j}^{t-1}\right) + r_{2,j} \cdot \left(gb_{i,j} - x_{i,j}^{t-1}\right). \tag{10}$$

3) *Confident Model:* Learning from $\mathbf{PB}_i$

$$v_{i,j}^t = w \cdot v_{i,j}^{t-1} + r_{1,j} \cdot \left(pb_{i,j} - x_{i,j}^{t-1}\right). \tag{11}$$

From the aforementioned discussion, we can see that a particle i selects its learning model by relying on the relative performance of $\mathbf{E}_i$, $\mathbf{PB}_i$, and $\mathbf{GB}$. When $\mathbf{E}_i$ is better than $\mathbf{GB}$, particle i will conduct the *diffident* model. In this case, particle i directly flies toward $\mathbf{E}_i$, which has the best fitness, regardless of other bias. The main primary objective of the particle is to speed up the convergence, and then enhance exploitation ability. In the *mild* model, $\mathbf{GB}$ and $\mathbf{E}_i$ offer more favorable performance than $\mathbf{PB}_i$. In this case, the particle i selects $\mathbf{GB}$ and $\mathbf{E}_i$ as two exemplars, which are beneficial for exploitation and exploration, respectively. Thus, particle i can achieve a balance between exploration and exploitation through the *mild* model. While the newly generated $\mathbf{E}_i$ is worse than $\mathbf{PB}_i$, particle i only chooses $\mathbf{PB}_i$ as its learning exemplar (i.e., the *confident* model) due to the $\mathbf{PB}_i$ having very favorable performance. This learning model causes the particle to ramble near its historical best position $\mathbf{PB}_i$, which is beneficial for the exploration capability of the entire population.

Based on the multiple learning models, particles in the population can display different search behaviors, and then satisfy distinct requirements. To verify the performance of the learning models, a set of experiments is conducted in this article, the results of which are presented in Table S-VI and Fig. S-2 in the supplementary material. From the results, we can see that the multiple learning models give particles various and promising characteristics on different evolutionary stages.

According to the aforementioned discussions, the pseudocode of selecting the learning model is detailed as Algorithm 2.

Algorithm 2 Select_learning_model ($\mathbf{X}_i^t$, $\mathbf{V}_i^t$, $\mathbf{E}_i$, $\mathbf{PB}_i$, $\mathbf{GB}$)

```
Evaluate E_i;
If f(E_i) < f(GB)
    Update V_i^{t+1} according to Eq. (9); GB = E_i;
Else If f(E_i) < f(PB_i)
    Update V_i^{t+1} according to Eq. (10);
Else
    Update V_i^{t+1} according to Eq. (11);
End If
Update X_i^{t+1} according to Eq. (2);
Output results: X_i^{t+1}, V_i^{t+1} and f(E_i).
```

D. Reusing Exemplars From the Outstanding Archive

While $\mathbf{E}_i$ is better than $\mathbf{GB}$ or $\mathbf{PB}_i^t$, it will be selected as a real exemplar and then provides much helpful knowledge for a particle i. However, there are two issues that need to be considered. One issue is that $\mathbf{E}_i$ may not supply enough positive information for the particle i in only one generation due to the stochastic factors in the learning model. Another issue is that $\mathbf{E}_i$ may be able to offer helpful knowledge for other particles even though it cannot help the particle i achieve a positive improvement. Thus, it is a conservative and economical choice that saving $\mathbf{E}_i$ in an archive can be reused by other particles.

Denoting A_o as a set of archived exemplars, the initialization and update of A_o are made very simple to avoid significant computational overhead. The maximum size of A_o is the same as the population size (N), and it is initiated to be empty. In each generation t, only if a new generated $\mathbf{E}_i$ is better than $\mathbf{PB}_i^t$, will it be added to A_o. While the archive size of A_o exceeds N, some solutions need to be removed from A_o, aiming to keep the size of A_o at N. Generally, there are two common methods for the update operator. The one method is using $\mathbf{E}_i$ to replace the worst individual in A_o. The main shortcoming of the method is that population diversity in A_o may be rapidly lost. The other one is replacing a randomly selected solution from A_o by $\mathbf{E}_i$. In this condition, the best solution in A_o may be replaced by $\mathbf{E}_i$, which is harmful to efficiently use those outstanding exemplars. Thus, in this article, a tournament strategy is adopted to update A_o. Specifically, an inferior solution of two randomly selected solutions from A_o is replaced by $\mathbf{E}_i$. In other words, the lost particle of the tournament is removed from A_o while a favorable $\mathbf{E}_i$ is added into A_o.

How to reuse A_o also needs to be carefully considered. In this article, some inferior particles in the current population take advantage of the outstanding solutions in A_o to improve their search capabilities. Like the update operation, the tournament strategy is also applied in the reuse process. Specifically, all solutions in A_o are randomly divided into $|A_o|/2$ groups, where $|A_o|$ is the size of A_o. Then each winner, who consists of better fitness of each group, is selected to replace an inferior particle in the current population. The process of reuse exemplars can be detailed as Algorithm 3.

To analyze the performance of A_o, A_e, and A_p, a set of experiments is conducted in this article, and the results are presented in Table S-VII in the supplementary material. The experimental results verify that A_o and A_e are beneficial for enhancing exploitation ability, while A_p plays a positive role on the balance of exploitation and exploration.

Algorithm 3 Reuse_exemplars (A_o, $\mathbf{PB}_i^t$ $(1 \le i \le N)$)

```
Sort all PB_i^t from worse to better, in terms of fitness;
Divide the solutions in A_o into |A_o|/2 groups; /*|A_o| is the size of A_o*/
For i = 1 to |A_o|/2 Do
    E = the better solution in the group i;
    If f(E)<f(PB_{j_i}^t)
        PB_{j_i}^t = E; /* j_i is the index of PB after sorting */
    End If
End For
Output result: PB_i^t (1 ≤ i ≤ N).
```

Algorithm 4 TAPSO

```
/* Initialization */
Initialize population size N, t = 04, A_e^t = A_p^t = A_o = ∅, p_c, and p_m;
For i = 1 to N Do
    Randomly initialize V_i^t, and X_i^t; Evaluate X_i^t; PB_i^t = X_i^t;
End For
Update GB;
/* Main Loop */
While not meeting terminal conditions
    t = t + 1;
    Update A_e^t and A_p^t based on Eq. (4) and Eq. (5);
    Calculate p_k for each candidate k in A_e^t and A_p^t according to Eq. (6);
    For i = 1 to N Do
        Generate E_i according to Algorithm 1;
        Carry out the learning process according to Algorithm 2;
        If |A_o^t| < N and f(E_i) < f(PB_i^t) /* Update archive A_o^t */
            A_o^t = A_o^t ⋃ E_i;
        Else If |A_o^t| ≥ N and f(E_i) < f(PB_i^t)
            Randomly select E_{j_1} and E_{j_2} from A_o;
            Use E_i to replace the poorer one between E_{j_1} and E_{j_2};
        End If
        Evaluate X_i^t and update PB_i^t;
    End For
    Reuse the archive A_o^t according to Algorithm 3;
    Update GB;
End While
Output results.
```

E. Framework of TAPSO

By incorporating the aforementioned components, the pseudocode of TAPSO is shown in Algorithm 4, and the flowchart of it is sketched in Fig. S-1 of the supplementary material.

IV. Experimental Studies

A. Benchmark Functions and Peer Algorithms

In this article, 30 benchmark functions, that is, 4 basic unimodal problems (f_1–f_4), 4 modified unimodal problems (f_5–f_8), 12 basic multimodal problems (f_9–f_{20}), and 10 modified multimodal problems (f_{21}–f_{30}), are selected to testify the performance of TAPSO on different environments. The experiments are conducted on dimensions of $D = 10$, 30, and 50. The maximum number of function evaluations is set to $MaxFEs$=10 000×D. Due to space limitations, the basic information of the benchmark functions is given in Table S-I of the supplementary material, and the details of the functions can refer to [9], [13], [37], [45], and [58].

For comparative analysis, other eight state-of-the-art PSO variants, including F_PSO [59], OLPSO [10], SLPSO [13], PSODDS [60], HCLPSO [33], CCPSO-ISM [61], SRPSO [22], and EPSO [48], are selected as peer algorithms in this article. Parameters settings of all peer algorithms are summarized in Table I. Note that population sizes for each algorithm in Table I, including three integers, denote the three population sizes for $D = 10$, 30, and 50, respectively. To obtain statistical results, each algorithm is carried out by 30 independent runs on each benchmark function. All of the algorithms are coded in MATLAB and run on a PC with an Intel Core i5-4200U CPU@1.6 GHz/4 GB RAM. (Note that only a single processor is used.)

TABLE I
Basic Information of Nine Peer Algorithms

Algorithm	Population size	Parameters Settings
*F_PSO	20 / 40 / 60	w=0.9∼0.4, $T_{max} = 3 * N^2$, $K = 4 * N$
OLPSO	20 / 40 / 60	w=0.9∼0.4,c=2.0,G=5
SLPSO	10 / 20 / 30	w=0.9∼0.4,η=1.496,γ=0.01
PSODDS	20 / 40 / 60	χ=0.7298, c_1=c_2=2.05
HCLPSO	20 / 40 / 60	w=0.99∼0.2, c_1=2.5∼0.5, c_2=0.5∼2.5, c=3∼1.5
CCPSO-ISM	20 / 40 / 60	P=0.05, G=5, c=2.0, w=0.6
SRPSO	20 / 40 / 60	w_I=1.05, w_F=0.5, c_1=c_2=1.49445
#EPSO	20 / 40 / 60	-
TAPSO	20 / 40 / 60	w=0.7298, p_c=0.5, p_m=0.02, M=N/4

* To describe simply, Frankenstein's PSO is renamed as F_PSO in this study.
\# There are too many parameters involved in EPSO. Due to the space limitation, more detailed information of the parameters setting one can refer to the corresponding literature.

Moreover, extensive experiments are also conducted in this article that aim to analyze the performance of new introduced strategies and parameters. The results of the experiments are presented in the supplementary material.

B. Solutions Accuracy

In this article, mean value (*Mean*), standard deviation (*Std.Dev.*), rank of mean value (*Rank*), and two-tailed t-test results are four basic performance metrics. Note that freedom at a 0.05 level of significance is adopted in the t-test.

The comparison results on solutions accuracy are provided in this section. Due to space limitations, only the results of 10-D are presented in Tables II–IV while the results of 30-D and 50-D are listed in Tables S-V–S-X of the supplementary material, where the best result of *Mean* on each function is shown in bold. Moreover, the *Rank* and the t-test results are also included in these tables. Note that the results of the t-test presented in the tables are "+," "−," and "=," which denote that TAPSO is significantly better than, significantly worse than, and almost the same as the corresponding competitor algorithms, respectively.

1) Unimodal Functions (f_1–f_8): From Table II and Tables S-III and S-XI in the supplementary material, we can see that TAPSO achieves the most favorable performance on the unimodal functions both in 10-D, 30-D, and 50-D, in terms of *Mean* and *Rank*. Specifically, TAPSO attains the best mean values on 7, 5, and 5 out of the eight unimodal functions in 10-D, 30-D, and 50-D, respectively. The t-test results manifest that TAPSO overwhelmingly dominates other peer algorithms on f_1, f_2, f_6, and f_8 with the 3-D cases while it offers the most promising performance on f_3 in both 10-D and 50-D cases. Furthermore, TAPSO attains noninferior performance to other peer algorithms on f_6 and f_8 in the 3-D cases. Although TAPSO offers very favorable results on f_6 in the three cases, it does not yield promising solutions on f_7, which is a variant of f_6 by adding some noise into fitness. In contrast, F_PSO,

TABLE II
OPTIMIZATION RESULTS ON THE EIGHT UNIMODAL FUNCTIONS ($D = 10$)

Algorithms	f_1				f_2				f_3				f_4			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	1.21E-148	5.11E-148	2	+	2.21E-81	3.29E-81	2	+	9.05E-28	3.03E-27	4	+	**0.00E+00**	0.00E+00	1	=
OLPSO	8.09E-61	1.57E-60	6	+	4.05E-34	6.40E-34	6	+	1.07E+02	1.54E+02	9	+	**0.00E+00**	0.00E+00	1	=
SLPSO	1.50E-99	8.22E-99	3	+	2.21E-44	8.82E-44	4	+	1.01E-01	2.18E-01	7	+	8.01E+02	1.11E+03	9	+
PSODDS	2.31E-92	1.03E-91	4	+	1.43E-46	1.82E-46	3	+	1.41E-82	3.88E-82	2	+	1.46E-12	2.21E-12	5	+
HCLPSO	6.81E-45	2.07E-44	8	+	4.01E-27	8.85E-27	7	+	1.56E-05	3.83E-05	6	+	1.22E+00	1.49E+00	7	+
CCPSO-ISM	4.80E-27	1.67E-26	9	+	1.56E-19	2.07E-19	9	+	2.05E+01	1.30E+01	8	+	7.42E+00	8.91E+00	8	+
SRPSO	7.17E-81	2.56E-80	5	+	2.54E-43	7.25E-43	5	+	6.55E-29	2.11E-28	3	+	**0.00E+00**	0.00E+00	1	=
EPSO	3.51E-46	9.53E-46	7	+	1.01E-24	1.71E-24	8	+	1.73E-24	3.34E-24	5	+	2.83E-03	1.50E-02	6	+
TAPSO	**4.48E-273**	0.00E+00	1		**1.99E-138**	9.77E-138	1		**5.16E-132**	1.98E-131	1		**0.00E+00**	0.00E+00	1	
	f_5				f_6				f_7				f_8			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	8.00E-29	2.56E-29	5	+	1.40E-26	2.40E-26	2	+	**2.67E-08**	6.54E-08	1	-	9.14E+04	5.85E+04	3	+
OLPSO	6.88E+00	2.62E+01	8	+	2.02E+02	1.12E+02	9	+	2.84E+02	1.63E+02	7	+	3.46E+06	4.93E+06	9	+
SLPSO	1.05E-31	5.76E-31	4	+	4.90E-01	9.31E-01	5	+	1.37E+03	1.98E+03	9	+	4.38E+05	5.13E+05	7	+
PSODDS	1.99E-28	1.48E-28	7	+	2.47E+01	4.90E+01	6	+	6.00E+01	8.53E+01	4	-	1.10E+05	1.99E+05	5	+
HCLPSO	**0.00E+00**	0.00E+00	1	=	4.73E-06	7.53E-06	4	+	9.57E-03	1.64E-02	3	-	1.12E+05	7.96E+04	6	+
CCPSO-ISM	**0.00E+00**	0.00E+00	1	=	1.16E+02	6.99E+01	8	+	4.01E+02	2.94E+02	8	=	7.21E+05	3.96E+05	8	+
SRPSO	1.38E+01	3.57E+01	9	+	5.31E+01	8.42E+01	7	+	7.68E+01	2.88E+02	5	=	1.09E+05	1.04E+05	4	+
EPSO	1.50E-28	1.55E-28	6	+	1.21E-24	1.51E-24	3	+	1.58E-07	8.05E-07	2	-	5.20E+04	3.76E+04	2	=
TAPSO	**0.00E+00**	0.00E+00	1		**4.99E-28**	3.79E-28	1		2.00E+02	3.81E+02	6		**3.14E+04**	3.34E+04	1	

TABLE III
OPTIMIZATION RESULTS ON THE 12 BASIC MULTIMODAL FUNCTIONS ($D = 10$)

Algorithms	f_9				f_{10}				f_{11}				f_{12}				f_{13}				f_{14}			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	**3.55E-15**	0.00E+00	1	=	7.14E+02	2.71E+02	7	+	2.48E-01	4.24E-01	7	+	1.55E+00	1.07E+00	7	+	**0.00E+00**	0.00E+00	1	-	**4.71E-32**	0.00E+00	1	=
OLPSO	**3.55E-15**	0.00E+00	1	=	4.74E+01	6.69E+01	5	+	**0.00E+00**	0.00E+00	1	=	2.00E-01	4.84E-01	6	+	**0.00E+00**	0.00E+00	1	-	**4.71E-32**	0.00E+00	1	=
SLPSO	6.39E-15	2.36E-15	7	+	1.58E+01	4.09E+01	4	+	3.35E-02	1.82E-01	6	+	9.26E-06	4.47E-05	4	+	1.18E-16	6.49E-16	6	-	2.07E-02	7.89E-02	9	+
PSODDS	6.39E-15	2.17E-15	7	+	8.75E+02	2.66E+02	8	+	7.89E+00	4.56E+00	9	+	5.43E+00	1.91E+00	9	+	1.50E-01	6.04E-01	9	+	4.78E-32	2.89E-33	6	+
HCLPSO	4.50E-15	1.60E-15	5	+	3.95E+00	2.16E+01	2	+	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	-	**4.71E-32**	0.00E+00	1	=
CCPSO-ISM	9.12E-15	2.90E-15	9	+	2.60E+02	1.18E+02	6	+	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	-	4.83E-32	3.96E-33	7	+
SRPSO	6.28E-15	1.79E-15	6	+	8.78E+02	2.92E+02	9	+	6.91E+00	2.13E+00	8	+	3.10E+00	1.32E+00	8	+	4.74E-16	1.23E-15	7	-	**4.71E-32**	0.00E+00	1	=
EPSO	4.38E-15	1.53E-15	4	+	1.18E+01	4.77E+01	3	+	**0.00E+00**	0.00E+00	1	=	3.33E-02	1.83E-01	5	+	**0.00E+00**	0.00E+00	1	-	5.14E-32	1.01E-32	8	+
TAPSO	**3.55E-15**	0.00E+00	1		**0.00E+00**	0.00E+00	1		**0.00E+00**	0.00E+00	1		**0.00E+00**	0.00E+00	1		1.62E-03	6.08E-03	8		**4.71E-32**	0.00E+00	1	
	f_{15}				f_{16}				f_{17}				f_{18}				f_{19}				f_{20}			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	**9.99E-02**	3.06E-10	3	-	1.65E+00	2.29E-01	7	=	8.97E-01	5.94E-01	3	+	4.28E-02	3.87E-02	6	=	1.30E+00	3.16E-01	9	+	9.12E+02	2.58E+03	6	+
OLPSO	6.17E-01	4.35E-01	9	+	1.07E+00	2.05E-01	6	+	5.07E+00	6.96E+00	7	+	1.14E-02	1.37E-02	3	-	6.00E-01	1.59E-01	6	+	2.57E+02	5.02E+02	5	+
SLPSO	4.87E-01	3.03E-01	8	+	5.29E-01	3.60E-01	2	=	2.62E+01	7.88E+01	8	+	3.09E-02	2.68E-02	5	=	3.97E-01	1.46E-01	3	=	1.37E+02	3.89E+02	4	+
PSODDS	3.93E-01	3.40E-01	6	+	1.66E+00	5.00E-01	8	+	2.03E+02	7.66E+02	9	+	1.24E-01	8.24E-02	9	+	8.01E-01	3.07E-01	8	+	2.07E+03	5.97E+03	9	+
HCLPSO	**9.99E-02**	4.85E-11	2	-	**4.57E-01**	2.33E-01	1	=	1.01E+00	1.19E+00	5	+	9.85E-03	8.81E-03	2	-	3.68E-01	8.51E-02	2	=	2.32E+01	1.88E+01	2	+
CCPSO-ISM	1.87E-01	5.07E-02	4	-	7.06E-01	3.39E-01	3	=	9.22E-01	1.14E+00	4	+	**8.89E-03**	7.92E-03	1	-	4.28E-01	1.48E-01	4	+	9.20E+02	8.20E+02	7	+
SRPSO	4.73E-01	5.40E-01	7	+	1.88E+00	4.23E-01	9	+	1.35E+00	1.51E+00	6	+	6.75E-02	2.66E-02	8	=	6.71E-01	2.35E-01	7	+	1.82E+03	3.36E+03	8	+
EPSO	**9.99E-02**	2.79E-11	1	-	9.39E-01	3.07E-01	5	=	1.37E-01	1.84E-01	2	+	2.18E-02	1.24E-02	4	-	4.96E-01	1.03E-01	5	=	2.47E+01	5.46E+01	3	+
TAPSO	2.70E-01	7.94E-02	5		7.96E-01	2.35E-01	4		**1.42E-11**	6.49E-11	1		5.98E-02	3.21E-02	7		**3.64E-01**	7.76E-02	1		**1.04E+01**	9.16E+00	1	

SRPSO, and EPSO demonstrate very promising performance on f_7 in the 3-D cases. The comparison results indicate that the performance of TAPSO on noise functions needs further improvement.

Along with the dimension increasing, TAPSO not only displays favorable performance on those simple unimodal problems, such as f_1, f_2, and f_3, but also demonstrates very reliable performance on the other nonseparable problems, including f_6 and f_8, measured by *Mean* and *Rank*. Thus, we can say that TAPSO has better scalability as well as higher solutions accuracy.

2) Basic Multimodal Functions (f_9–f_{20}): The results in Table III and Tables S-IX and S-XII in the supplementary material indicate that TAPSO outperforms other peer algorithms on the basic multimodal functions in the 3-D cases, in terms of the number of attained best results on *Mean* and *Rank*. Moreover, the *t*-test results demonstrate that TAPSO attains significantly better than or almost the same as all of the other eight peer algorithms on at least 7 out of the 12 multimodal functions in 10-D and 30-D cases. In the 50-D case, this number decreases to 6. In addition, TAPSO is the only one who obtains the global best optimum of f_{12} on all 30 runs in all of the dimension cases. On the contrary, although HCLPSO and CCPSO-ISM yield the same results as TAPSO on f_{12} in 10-D cases, the performance of the two peer algorithms has rapidly deteriorated along with increasing dimensions. Moreover, TAPSO also attains very promising results on f_{11} while HCLPSO attains favorable performance on f_{18} in the three different dimension cases. Thus, we can obtain a conservative conclusion that multiple learning models (i.e., *diffident*, *mild*, and *confident* models) in TAPSO can provide richer information to deal with different search objectives than a single learning model.

3) Shifted/Rotated Multimodal Functions (f_{21}–f_{30}): The comparison results on the ten shifted/rotated multimodal functions in the 3-D cases are summarized in Table IV and Tables S-X and S-XIII in the supplementary material, respectively. The experimental results indicate that HCLPSO offers the most favorable performance on the modified multimodal functions in the 10-D case, in terms of *Mean* and *Rank*. Its performance, however, dramatically deteriorates with the increase of dimension. On the contrary, the improvements of TAPSO become more salient with the increase of dimension. For instance, TAPSO is dominated by HCLPSO in 6 out of the 10 functions in 10-D case. However, TAPSO yields better results than HCLPSO as well as other peer algorithms in 30-D and 50-D cases. Thus, we can say that TAPSO has very reliable scalability on these modified multimodal functions. Together, with the results on the 12 basic multimodal functions, we can find out that the different learning models applied in TAPSO are favorable for multimodal functions.

C. Statistical Results of Solutions

1) t-Test Results: The results of the *t*-test between TAPSO and other peer algorithms are among the 30 functions in

TABLE IV
OPTIMIZATION RESULTS ON THE TEN MODIFIED MULTIMODAL FUNCTIONS ($D = 10$)

Algorithms	f_{21}				f_{22}				f_{23}				f_{24}				f_{25}			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	7.11E-01	8.59E-01	7	+	1.46E+00	1.10E+00	7	+	1.62E+00	1.60E+00	4	+	2.92E+00	3.21E-01	3	=	1.88E-01	1.04E-01	2	=
OLPSO	6.63E-02	2.52E-01	5	+	4.67E-01	6.29E-01	6	+	1.03E+02	1.95E+02	7	+	3.85E+00	1.75E-01	9	+	1.35E+00	3.10E+00	8	+
SLPSO	6.63E-02	2.52E-01	6	+	3.75E-06	1.39E-05	5	+	6.19E+01	9.37E+01	6	+	3.08E+00	4.67E-01	4	=	3.89E-01	4.34E-01	6	+
PSODDS	9.88E+00	5.25E+00	9	+	8.90E+00	5.57E+00	9	+	4.40E+02	2.05E+03	8	+	3.66E+00	3.03E-01	8	=	7.12E-01	4.67E-01	7	+
HCLPSO	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	-	6.55E-01	1.04E+00	2	+	**2.60E+00**	3.31E-01	1	=	**9.89E-02**	4.52E-02	1	-
CCPSO-ISM	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	-	1.18E+01	1.86E+01	5	+	3.49E+00	2.55E-01	7	+	2.82E-01	1.50E-01	4	=
SRPSO	6.07E+00	2.77E+00	8	+	3.57E+00	1.50E+00	8	+	7.20E+05	2.87E+06	9	+	2.65E+00	5.10E-01	2	=	1.76E+00	3.45E+00	9	+
EPSO	**0.00E+00**	0.00E+00	1	=	**0.00E+00**	0.00E+00	1	-	7.35E-01	1.75E+00	3	+	3.29E+00	5.69E-01	6	+	2.97E-01	1.64E-01	5	=
TAPSO	**0.00E+00**	0.00E+00	1		2.37E-16	6.14E-16	4		**3.44E-07**	1.65E-06	1		3.28E+00	4.38E-01	5		2.63E-01	2.08E-01	3	
	f_{26}				f_{27}				f_{28}				f_{29}				f_{30}			
	Mean	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test	*Mean*	*Std.Dev*	*Rank*	*t*-test
F_PSO	2.04E+01	7.01E-02	8	+	1.28E+01	4.62E+00	4	=	1.02E+01	2.26E+00	3	-	1.42E+00	1.24E+00	2	=	**9.99E-02**	3.35E-10	3	-
OLPSO	2.05E+01	1.18E-01	9	+	2.40E+01	1.08E+01	7	+	2.21E+01	6.53E+00	7	=	5.40E+00	1.23E+00	6	+	5.00E-01	4.02E-01	8	+
SLPSO	2.01E+01	6.49E-02	3	+	1.54E+01	7.48E+00	5	=	1.44E+01	6.87E+00	5	=	5.50E+00	1.04E+00	7	+	3.97E-01	1.63E-01	6	+
PSODDS	**2.00E+01**	3.25E-02	2	=	2.77E+01	1.05E+01	8	+	2.72E+01	1.26E+01	8	=	5.50E+00	1.44E+00	8	=	7.33E-01	6.12E-01	9	+
HCLPSO	2.03E+01	8.15E-02	5	+	**6.74E+00**	2.86E+00	1	-	9.33E+00	2.02E+00	2	-	3.05E+00	9.24E-01	3	-	**9.99E-02**	1.01E-10	2	-
CCPSO-ISM	2.01E+01	4.53E-02	4	+	5.48E+01	1.56E+01	9	+	4.32E+01	9.07E+00	9	=	6.56E+00	7.53E-01	9	+	1.87E-01	3.46E-02	4	-
SRPSO	2.04E+01	6.47E-02	7	+	9.23E+00	3.81E+00	3	=	**8.28E+00**	1.57E+00	1	-	**1.14E+00**	1.05E+00	1	-	4.00E-01	4.66E-01	7	+
EPSO	2.03E+01	7.81E-02	6	+	7.40E+00	3.03E+00	2	-	1.06E+01	1.87E+00	4	-	4.12E+00	6.70E-01	4	=	**9.99E-02**	7.52E-11	1	-
TAPSO	**2.00E+01**	3.15E-02	1		1.63E+01	5.45E+00	6		2.12E+01	1.19E+01	6		4.27E+00	1.60E+00	5		2.63E-01	7.65E-02	5	

TABLE V
t-TEST RESULTS BETWEEN TAPSO AND OTHER PSO VARIANTS ON ALL TEST FUNCTIONS

Algorithm	10D			30D			50D			*CP*
	#+	#=	#-	#+	#=	#-	#+	#=	#-	
F_PSO	16	9	5	20	3	7	23	2	5	42
OLPSO	23	5	2	22	8	0	24	3	3	64
SLPSO	23	6	1	19	10	1	19	7	4	55
PSODDS	25	4	1	24	5	1	27	2	1	73
HCLPSO	12	8	10	13	12	5	15	9	6	19
CCPSO-ISM	17	8	5	17	8	5	24	2	4	44
SRPSO	21	6	3	21	8	1	22	5	3	57
EPSO	15	7	8	17	8	5	17	8	5	31

TABLE VI
FRIEDMAN-TEST OF MEAN VALUES ON THE 30 TEST FUNCTIONS

Overall Rank	Algorithm	Ranking	10D Algorithm	10D Ranking	30D Algorithm	30D Ranking	50D Algorithm	50D Ranking
1	TAPSO	2.75	TAPSO	3.08	TAPSO	2.57	TAPSO	2.53
2	HCLPSO	3.28	HCLPSO	3.23	HCLPSO	3.40	HCLPSO	3.22
3	EPSO	3.72	EPSO	4.07	EPSO	3.58	EPSO	3.50
4	SLPSO	4.93	F_PSO	4.17	SLPSO	4.87	SLPSO	4.35
5	F_PSO	5.21	SLPSO	5.60	FPSO	5.05	OLPSO	6.00
6	CCPSO_ISM	5.61	CCPSO_ISM	5.60	CCPSO_ISM	5.20	SRPSO	6.03
7	SRPSO	6.03	SRPSO	6.07	SRPSO	5.98	CCPSO_ISM	6.05
8	OLPSO	6.52	OLPSO	6.23	PSODDS	7.03	FPSO	6.42
9	PSODDS	6.96	PSODDS	6.95	OLPSO	7.32	PSODDS	6.90

TABLE VII
SCORES AND FINAL RANK OF ALL PEER ALGORITHMS

Algorithms	*Score1*	*Score2*	*Score*	*FR*
F_PSO	1.50	22.52	24.02	5
OLPSO	0.32	19.50	19.82	9
SLPSO	7.01	26.42	33.43	4
PSODDS	3.65	17.81	21.46	8
HCLPSO	14.94	39.32	54.26	3
CCPSO-ISM	1.02	22.10	23.12	6
SRPSO	0.56	20.60	21.16	7
EPSO	44.99	34.70	79.69	2
TAPSO	50.00	50.00	100.00	1

the 3-D cases. In this section, statistical results of the *t*-test are displayed in Table V, in which symbols "#+," "#−," and "#=" denote the number that TAPSO is significantly better than, significantly worse than, and almost the same as the corresponding competitor algorithm, respectively. The comprehensive performance (CP) is equal to "#+" minus "#−."

It can be seen from Table V that TAPSO significantly outperforms the other eight PSO variants on the majority of test functions. Concretely, TAPSO offers more favorable performance than the other eight peer algorithms both on the 10-D, 30-D, and 50-D cases. Moreover, according to the values of CP, we can see TAPSO displays the most promising performance, followed by HCLPSO and EPSO.

2) Friedman-Test Results: In this part, a Friedman-test of *Mean* values is used to compare the overall performance among all nine competitors. Furthermore, we also separately carry out the test on the 10-D, 30-D, and 50-D cases. The results are listed in Table VI, in which each algorithm and its rankings are listed in ascending order (the lower the better).

From the results, we can see that TAPSO attains the best overall performance, followed by HCLPSO and EPSO, which is consistent with the *t*-test results listed in Table V. In addition, TAPSO also achieves the most favorable results on both the three different dimension cases while HCLPSO and EPSO yield the second- and the third-best performance, respectively. Although OLPSO does not offer promising performance in 10-D and 30-D cases, it shows many favorable characteristics in the 50-D case than in the two lower cases, which means that the orthogonal learning strategy in OLPSO may be very suitable for higher dimension problems.

D. Final Rank of All Peer Algorithms

To evaluate the overall performance of all peer algorithms and give a final rank (FR) of them, an evaluation method based on a score of 100 is applied in this article. There are two criteria involved in the evaluation method, in which higher weights are given for higher dimensions. Due to space limitations, two criteria and the entire evaluation are detailed in the supplementary material.

From the results of FR as well as *Score* displayed in Table VII, we can see that TAPSO offers the best overall performance measured by FR, while EPSO is the second best, followed by HCLPSO and SLPSO. Moreover, TAPSO not only attains the highest value of FR but also performs best results on *Score1* and *Score2*. From the superior performance of these PSO variants, we can obtain a conservative conclusion that assigning different roles for different particles (or subpopulations) is beneficial to improve the comprehensive performance of PSOs on different problems.

E. Convergence Speed

The speed in obtaining a global optimum is also a salient yardstick for measuring an algorithm. Due to space limitations,

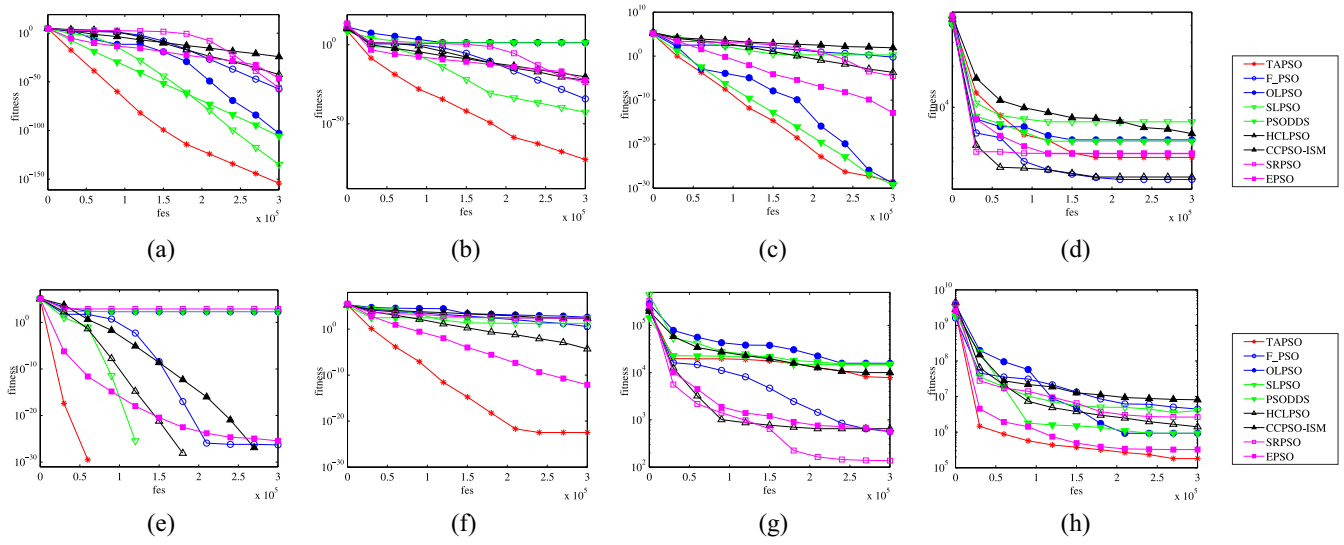


Fig. 2. Comparison results of convergence characteristics on the eight unimodal functions: (a) f_1, (b) f_2, (c) f_3, (d) f_4, (e) f_5, (f) f_6, (g) f_7, and (h) f_8.

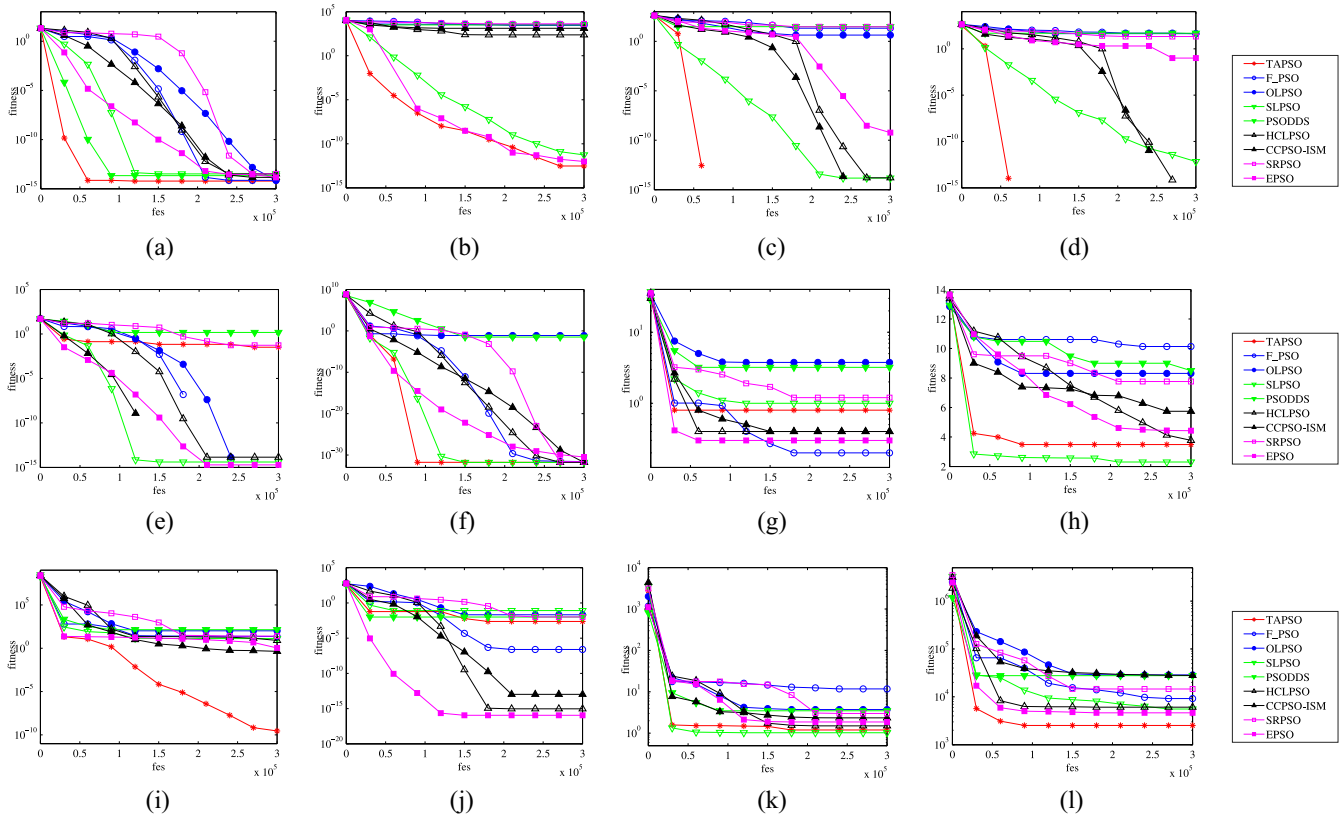


Fig. 3. Comparison results of convergence characteristics on the 12 basic multimodal functions: (a) f_9, (b) f_{10}, (c) f_{11}, (d) f_{12}, (e) f_{13}, (f) f_{14}, (g) f_{15}, (h) f_{16}, (i) f_{17}, (j) f_{18}, (k) f_{19}, and (l) f_{20}.

in this section, experiments are only conducted on the 30-D case to compare the convergence process. The convergence graphs of the three-class functions, that is, unimodal, basic multimodal, and modified multimodal functions, are shown in Figs. 2–4, respectively. Moreover, the experiments of time usages on the 30-D functions are also conducted in this article. The results of the experiments are listed in Table VIII.

From Fig. 2, we can see that TAPSO displays the highest convergence speed during the entire evolutionary process on 5 out of the 8 unimodal functions, and offers the most accurate solutions on the five functions. On f_1, f_2, f_5, and f_6, the convergence speed of TAPSO is significantly better than other algorithms while SRPSO yields the highest convergence speed at the later evolutionary stage on f_7. Furthermore, TAPSO performs slightly better than EPSO on f_8. Although there are four

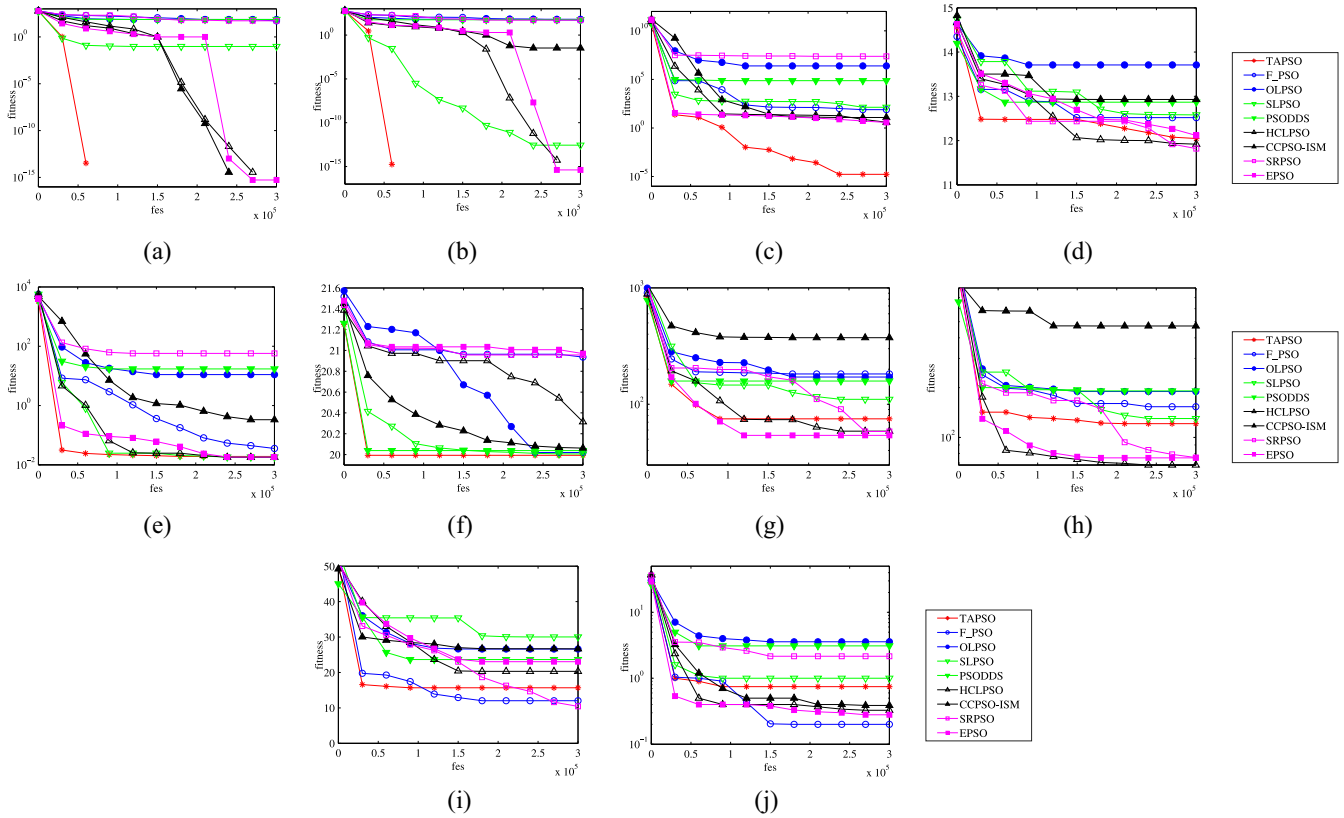


Fig. 4. Comparison results of convergence characteristics on the ten modified functions: (a) f_{21}, (b) f_{22}, (c) f_{23}, (d) f_{24}, (e) f_{25}, (f) f_{26}, (g) f_{27}, (h) f_{28}, (i) f_{29}, and (j) f_{30}.

TABLE VIII
COMPARISON OF TIME USAGES ON 30-D FUNCTIONS (IN SECOND)

	F_PSO	OLPSO	SLPSO	PSODDS	HCLPSO	CPSO-ISM	SRPSO	EPSO	TAPSO
f_1	134.90	122.25	116.33	129.05	128.55	135.10	133.68	123.94	127.33
f_2	139.33	127.71	119.24	130.89	130.02	119.96	137.83	126.00	129.43
f_3	154.17	143.85	137.95	145.29	143.93	137.86	150.78	142.67	142.22
f_4	28.77	22.34	20.77	27.52	23.14	27.09	27.10	36.51	32.12
f_5	28.02	22.30	23.38	27.09	22.74	25.21	27.17	30.07	29.73
f_6	40.54	39.83	34.46	41.46	38.61	42.09	43.08	45.12	40.06
f_7	48.43	42.00	40.75	47.38	44.47	48.65	48.87	52.21	47.93
f_8	32.91	26.12	24.29	31.40	28.06	30.93	30.71	34.92	31.22
f_9	139.31	131.67	124.65	130.67	132.60	121.78	141.57	127.52	130.36
f_{10}	133.93	130.79	119.25	127.68	130.16	120.24	146.52	131.53	130.28
f_{11}	134.95	130.31	117.89	128.18	127.52	119.49	145.63	124.91	135.34
f_{12}	136.93	131.56	119.72	128.08	130.50	121.07	147.82	126.49	139.31
f_{13}	229.24	220.50	207.92	213.66	217.27	206.73	244.44	209.65	214.47
f_{14}	145.43	138.79	129.27	135.19	137.85	126.36	153.55	131.87	141.33
f_{15}	26.84	21.47	17.00	26.35	20.96	25.64	23.01	28.99	27.38
f_{16}	280.03	272.94	248.62	259.05	271.09	257.79	247.95	263.72	260.74
f_{17}	133.77	128.89	117.28	126.06	129.00	119.66	141.51	124.71	132.64
f_{18}	136.89	134.96	122.46	128.01	128.62	121.43	116.35	126.90	136.16
f_{19}	50.10	46.27	40.41	48.97	45.62	50.01	56.89	53.09	50.11
f_{20}	35.40	31.05	26.80	34.09	30.13	34.20	38.71	46.40	38.36
f_{21}	29.16	22.73	20.85	26.59	23.27	27.36	30.69	31.22	30.24
f_{22}	30.31	25.37	20.25	27.39	24.50	28.39	31.97	32.44	31.52
f_{23}	30.09	25.76	22.30	29.47	25.94	29.12	33.30	32.23	30.28
f_{24}	57.43	52.32	45.70	53.38	54.26	56.73	60.24	61.15	57.24
f_{25}	32.10	27.83	23.24	30.24	28.04	29.58	33.49	34.87	31.84
f_{26}	31.78	27.50	22.88	29.64	28.85	31.16	34.26	42.31	31.27
f_{27}	30.53	23.80	19.94	28.11	25.01	28.34	30.60	34.11	29.74
f_{28}	31.20	25.76	21.21	28.93	27.46	30.07	33.28	36.01	30.21
f_{29}	110.63	105.32	96.50	104.59	106.36	110.08	122.42	112.21	109.20
f_{30}	30.35	26.57	21.71	28.78	26.32	29.43	32.22	33.28	29.79
Avg.	86.78	80.95	74.10	81.77	81.03	79.72	88.19	84.57	84.26

algorithms, including TAPSO, find out the global best solution on f_5, and TAPSO manifests the highest convergence speed than the other three competitors.

The results presented in Fig. 3 demonstrate that TAPSO attains the most outstanding performance on 6 out of the 12 basic multimodal functions, including f_9, f_{10}, f_{11}, f_{12}, f_{17}, and f_{20}. Although more than one algorithm attains the theoretical global optimal solutions on f_{11}, f_{12}, and f_{13}, TAPSO achieves the highest convergence speed on f_{11} and f_{12} while CCPSO-ISM offers the best performance on f_{13}. Moreover, TAPSO also displays the competitive results on f_{14}, f_{16}, and f_{19}, on which it performs slightly weaker than SLPSO. Note that all peer algorithms except TAPSO are trapped into local optimal on f_{17} since there is a very narrow valley from local optimum to global optimum within the fitness landscape. Thus, we can obtain a conservative conclusion that TAPSO has a very favorable exploration capability due to its high population diversity.

Although more than one algorithm obtains the global optimal solutions on f_{21} and f_{22}, TAPSO has the highest convergence speed, while HCLPSO also yields a very promising convergence process on the functions. In addition, TAPSO offers the most notable convergence performance besides the highest accurate solution on f_{23} while other peer algorithms fall into local optimum in the early evolution stage. Although TAPSO shows the fastest convergence on f_{24}–f_{29} at the initial stage, it cannot find out more accurate solutions on these functions at the later stage. Hence, we regard that the capability of jumping out of the local optimum of TAPSO needs to be further improved.

F. Time Usage

In this section, an experiment is conducted to compare the time usages of the nine competitors. The results listed in

TABLE IX
COMPARISON RESULTS ON FOUR REAL APPLICATIONS

		F-PSO	OLPSO	SLPSO	PSODDS	HCLPSO	CCPSO-ISM	SRPSO	EPSO	TAPSO
F_1	*Mean*	1.75E+01	1.61E+01	1.12E+01	1.80E+01	4.26E+00	1.61E+01	1.31E+01	**3.99E+00**	1.54E+01
	Best	8.42E+00	5.83E-10	4.17E-07	**0.00E+00**	4.19E-19	1.19E+00	**0.00E+00**	1.96E-13	**0.00E+00**
	Median	2.00E+01	1.79E+01	1.28E+01	1.86E+01	7.21E-04	1.73E+01	1.18E+01	**1.62E-05**	1.51E+01
F_2	*Mean*	9.25E-13	6.05E-13	8.97E-23	**0.00E+00**	8.19E-14	**0.00E+00**	3.88E-13	1.06E-12	**0.00E+00**
	Best	3.71E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.21E-21	**0.00E+00**	1.46E-19	2.19E-15	**0.00E+00**
	Median	1.02E-13	**0.00E+00**	3.93E-26	**0.00E+00**	2.62E-15	**0.00E+00**	3.79E-14	1.40E-13	**0.00E+00**
F_3	*Mean*	1.39E+00	1.80E+00	1.21E+00	1.16E+00	1.04E+00	1.10E+00	**9.61E-01**	9.89E-01	1.07E+00
	Best	9.38E-01	1.34E+00	1.00E+00	9.42E-01	8.68E-01	1.02E+00	**5.00E-01**	7.27E-01	7.31E-01
	Median	1.48E+00	1.81E+00	1.19E+00	1.20E+00	1.03E+00	1.08E+00	**8.86E-01**	9.92E-01	1.07E+00
F_4	*Mean*	-1.85E+01	-8.83E+00	-1.96E+01	-2.23E+01	-2.71E+01	-1.83E+01	-9.49E+00	-2.47E+01	**-2.72E+01**
	Best	-2.19E+01	-1.69E+01	-2.71E+01	-2.76E+01	**-2.84E+01**	-2.50E+01	-1.33E+01	-2.84E+01	-2.76E+01
	Median	-1.81E+01	-8.37E+00	-1.83E+01	-2.30E+01	-2.72E+01	-1.78E+01	-9.55E+00	-2.55E+01	**-2.75E+01**

TABLE X
FRIEDMAN-TEST RESULTS ON FOUR REAL APPLICATIONS

Overall	Mean		Best		Median	
Rank	Algorithm	Rank	Algorithm	Rank	Algorithm	Rank
1	TAPSO	3.00	TAPSO	2.88	TAPSO	3.13
2	HCLPSO	3.00	PSODDS	3.63	HCLPSO	3.25
3	EPSO	3.75	HCLPSO	3.88	EPSO	3.75
4	SLPSO	4.75	EPSO	4.38	SRPSO	4.75
5	SRPSO	4.75	SRPSO	4.75	SLPSO	5.00
6	CCPSO_ISM	5.13	SLPSO	5.50	CCPSO_ISM	5.13
7	PSODDS	5.25	CCPSO_ISM	6.25	PSODDS	5.38
8	FPSO	7.50	OLPSO	6.50	OLPSO	6.88
9	OLPSO	7.88	FPSO	7.25	FPSO	7.75

Table VIII indicate that SLPSO yields the best performance, followed by CCPSO-ISM and OLPSO. Although TAPSO offers the best performance measured by mean results (see Tables V and VI) and final rank (see Table VII), it manifests the sixth better performance in terms of the average time usage. Comparing TAPSO and canonical PSO, we can find out that it is the sorting operations in updating archives A_e and A_p that cause extra time consumption. However, together with the convergency curves demonstrated in Figs. 2–4, we can see that TAPSO also has relatively fast convergence speed.

G. Comparison on Real Applications

To testify the performance of TAPSO on real applications, four common engineering problems are adopted in this section, that is, F_1: parameter estimation for frequency-modulated (FM) sound waves, F_2: design of a gear train, F_3: spread spectrum radar polyphase code design, and F_4: Lennard–Jones potential problems. Due to space limitations, details of the four problems can be observed from [13] and [62].

The experimental results listed in Table IX include three performance metrics, that is, *Mean*, *Best*, and *Median* values of 30 independent runs. Furthermore, the results of three Friedman-tests, which are presented in Table X, are also, respectively, conducted on the three metrics.

From Table IX, we can see that TAPSO, PSODDS, and SRPSO yield the most favorable performance on F_1, in terms of *Best* value. On the contrary, EPSO and HCLPSO offer more promising *Median* results than the other seven peer algorithms. Although F_2 has been solved easily by all of the algorithms, TAPSO, PSODDS, and CCPSO-ISM achieve the most reliable performance than other algorithms since all three outstanding algorithms find out the global best solutions on all independent runs. On F_3, SRPSO displays the best performance on all performance metrics, followed by EPSO, HCLPSO, and TAPSO. Furthermore, TAPSO, HCLPSO, and EPSO yield more pleasurable characteristics than the other six peer algorithms, in terms of *Mean*, *Best*, and *Median* values. On F_4, TAPSO yields the most favorable performance measured by *Mean* and *Median* results, followed by HCLPSO.

The Friedman-test results listed in Table X show that TAPSO yields the best comprehensive performance among all nine peer algorithms. Moreover, HCLPSO and EPSO also offer very superior performance. Note that PSODDS displays the second-best performance, in terms of *Best* value, though it demonstrates unfavorable performance on the metrics *Mean* and *Median*.

V. CONCLUSION

In this article, inspired by a common phenomenon that areas with very steep surface curves are common around local optima, we proposed a TAPSO, in which the change of fitness in two consecutive generations (i.e., improvement rate) is regarded as an additional criterion when choosing exemplars for learner particles. In TAPSO, two archives are used to save profiteers particles and elite particles that offer greater improvement rates and better fitness, respectively. Based on ordinary genetic operators, two parents, respectively, selected from the two archives, are used to generate a potential exemplar for a particle. In this way, the bred exemplar is well diversified and highly qualified since it has characteristics with higher fitness as well as faster progress. In addition, those outstanding exemplars are saved in the third archive, which can be reused by particles to improve the exploitation and save computing resources. According to the performance of the potential exemplar, the particle adopts a proper learning model aiming to balance the exploration and exploitation capabilities. Moreover, to simplify the algorithm, acceleration coefficients applied in the canonical PSO are removed from TAPSO.

To verify the effectiveness of TAPSO, sets of extensive experiments were conducted. Furthermore, the performance of the three archives and the learning models was also analyzed by extensive experiments. From the comparison results, we can obtain some preliminary conclusions. First, elites and profiteers can provide different merits for other learner particles. Second, the new generated exemplar can help a particle find out a more promising position. Finally, different learning models have distinct characteristics, which is beneficial for satisfying different requirements of different fitness landscapes.

In TAPSO, an exemplar generated by an elite and a profiteer is used to provide various knowledge for a learner particle. Although elites and profiteers have their own characteristics, which are suitable for different problems, it is still an open question on how to design an appropriate strategy to breed exemplars according to a specific problem. Considering that many real applications are black-box problems, it is worth further studying that how to design an adaptive method to combine helpful information implied in different particles. Moreover, the advantages and disadvantages of the three learning models are also a promising research direction, in order to design a reliable and robust algorithm based on the heterogeneous multiple models [63] for different problems.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, WA, Australia, 1995, pp. 1942–1948.

[2] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[3] X. Zhang, K.-J. Du, Z.-H. Zhan, S. Kwong, T.-L. Gu, and J. Zhang, "Cooperative co-evolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2937565.

[4] M. Gong, Q. Cai, X. W. Chen, and L. J. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 82–97, Feb. 2014.

[5] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2933499.

[6] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.

[7] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proc. Swarm Intell. Symp.*, Indianapolis, IN, USA, Apr. 2003, pp. 174–181.

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

[9] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[10] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[11] R. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cogn. Sci. Soc.*, Amherst, MA, USA, Aug. 1986, pp. 823–831.

[12] X. W. Xia *et al.*, "An expanded particle swarm optimization based on multi-exemplar and forgetting ability," *Inf. Sci.*, vol. 508, pp. 105–120, Jan. 2020.

[13] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 627–646, Jun. 2012.

[14] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, Anchorage, AK, USA, May 1998, pp. 68–73.

[15] 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, Jun. 2004.

[16] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Seoul, South Korea, 2001, pp. 101–106.

[17] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Comput. Oper. Res.*, vol. 33, no. 3, pp. 859–871, Mar. 2006.

[18] J. Lu, H. Hu, and Y. Bai, "Generalized radial basis function neural network based on an improved dynamic particle swarm optimization and AdaBoost algorithm," *Neurocomputing*, vol. 152, pp. 305–315, Mar. 2015.

[19] C. Pornsing, M. S. Sodhi, and B. F. Lamond, "Novel self-adaptive particle swarm optimization methods," *Soft Comput.*, vol. 20, no. 9, pp. 3579–3593, Sep. 2016.

[20] A. A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3658–3670, Jun. 2011.

[21] L. M. Zhang, Y. G. Tang, C. C. Hua, and X. P. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques," *Appl. Soft Comput.*, vol. 28, pp. 138–149, Mar. 2015.

[22] M. R. Tanweer, S. Suresh, and N. Sundararajan, "Self regulating particle swarm optimization algorithm," *Inf. Sci.*, vol. 294, no. 10, pp. 182–202, Feb. 2015.

[23] M. R. Tanweer, S. Suresh, and N. Sundararajan, "Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems," *Inf. Sci.*, vol. 326, pp. 1–24, Jan. 2016.

[24] M. R. Tanweer, R. Auditya, S. Suresh, N. Sundararajan, and N. Srikanth, "Directionally driven self-regulating particle swarm optimization algorithm," *Swarm Evol. Comput.*, vol. 28, pp. 98–116, Jun. 2016.

[25] K. Yasuda and N. Iwasaki, "Adaptive particle swarm optimization using velocity information of swarm," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, The Hague, The Netherlands, 2004, pp. 3475–3481.

[26] J. Li, J. Q. Zhang, C. J. Jiang, and M. C. Zhou, "Composite particle swarm optimizer with historical memory for function optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2350–2363, Oct. 2015.

[27] H. G. Han, W. Lu, and J. F. Qiao, "An adaptive multiobjective particle swarm optimization based on multiple adaptive methods," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2754–2767, Sep. 2017.

[28] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, USA, May 2002, pp. 1671–1676.

[29] Q. Liu, W. Wei, H. Yuan, Z.-H. Zhan, and Y. Li, "Topology selection for particle swarm optimization," *Inf. Sci.*, vol. 363, pp. 154–173, Oct. 2016.

[30] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm Evol. Comput.*, vol. 39, pp. 24–35, Apr. 2018.

[31] G. Xu *et al.*, "Particle swarm optimization based on dimensional learning strategy," *Swarm Evol. Comput.*, vol. 45, pp. 33–51, Mar. 2019.

[32] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1567–1578, Sep. 2014.

[33] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.*, vol. 24, pp. 11–24, Oct. 2015.

[34] Q. D. Qin, S. Cheng, Q. Y. Zhang, L. Li, and Y. H. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2238–2251, Oct. 2016.

[35] A. S. Mohais, R. Mendes, C. Ward, and C. Posthoff, "Neighborhood restructuring in particle swarm optimization," in *Proc. Aust. Conf. Artif. Intell.*, Sydney, NSW, Australia, Dec. 2005, pp. 776–785.

[36] I. Hanaf, F. M. Cabrera, F. Dimane, and J. T. Manzanares, "Application of particle swarm optimization for optimizing the process parameters in turning of PEEK CF30 composites," in *Proc. 9th Int. Conf. Int. Eng.*, Târgu Mureş, Romania, Oct. 2016, pp. 195–202.

[37] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proc. IEEE Symp. Swarm Intell.*, Pasadena, CA, USA, 2005, pp. 124–129.

[38] Z.-H. Zhan, J. Li, J. Cao, J. Zhang, H. S.-H. Chung, and Y.-H. Shi, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[39] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[40] R. Cheng and Y. C. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[41] Z. Ren, A. M. Zhang, C. Y. Wen, and Z. R. Feng, "A scatter learning particle swarm optimization algorithm for multimodal problems," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1127–1140, Jul. 2014.

[42] Y.-J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[43] H. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Proc. IEEE Symp. Swarm Intell.*, Indianapolis, IN, USA, 2003, pp. 72–79.

[44] X. W. Xia, C. W. Xie, B. Wei, Z. B. Hu, B. J. Wang, and C. Jin, "Particle swarm optimization using multi-level adaptation and purposeful detection operators," *Inf. Sci.*, vols. 385–386, pp. 174–195, Apr. 2017.

[45] X. W. Xia, J. N. Liu, and Z. B. Hu, "An improved particle swarm optimizer based on tabu detecting and local learning strategy in a shrunk search space," *Appl. Soft Comput.*, vol. 23, no. 5, pp. 76–90, Oct. 2014.

[46] H. Haklı and H. Uğuz, "A novel particle swarm optimization algorithm with Levy flight," *Appl. Soft Comput.*, vol. 23, no. 5, pp. 333–345, Oct. 2014.

[47] E. Grasso, G. Di Bella, and C. Borean, "Ranked particle swarm optimization," *Int. J. Adv. Syst. Meas.*, vol. 8, nos. 1–2, pp. 18–29, Jun. 2015.

[48] N. Lynn and P. N. Suganthan, "Ensemble particle swarm optimizer," *Appl. Soft Comput.*, vol. 55, pp. 533–548, Jun. 2017.
[49] E. Bengoetxea and P. Larrañaga, "EDA-PSO: A hybrid paradigm combining estimation of distribution algorithms and particle swarm optimization," in *Proc. Int. Conf. Swarm Intell.*, Brussels, Belgium, 2010, pp. 416–423.
[50] B. Xin, J. Chen, J. Zhang, H. Fang, and Z.-H. Peng, "Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 5, pp. 744–767, Sep. 2012.
[51] M. S. Kiran and M. Gündüz, "A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2188–2203, Apr. 2013.
[52] X. Xu, Z. W. Shi, and B. Pan, "ℓ_0-based sparse hyperspectral unmixing using spectral information and a multi-objectives formulation," *ISPRS J. Photogrammetry Remote Sens.*, vol. 141, pp. 46–58, Jul. 2018.
[53] B. Pan, Z. W. Shi, and X. Xu, "Multi-objective based sparse representation classifier for hyperspectral imagery using limited samples," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 1, pp. 239–249, Jan. 2019.
[54] R. Rosales, R. A. Rehfeldt, and S. Lovett, "Effects of multiple exemplar training on the emergence of derived relations in preschool children learning a second language," *Anal. Verbal Behav.*, vol. 27, no. 1, pp. 61–74, 2011.
[55] K. E. Twomey, S. L. Ranson, and J. S. Horst, "That's more like it: Multiple exemplars facilitate word learning," *Infant Child Dev.*, vol. 23, no. 2, pp. 105–122, 2014.
[56] A. Baeck, K. Maes, C. Van Meel, and H. P. O. de Beeck, "The transfer of object learning after training with multiple exemplars," *Front. Psychol.*, vol. 7, p. 1386, Sep. 2016.
[57] P. Yang, K. Tang, and X. F. Lu, "Improving estimation of distribution algorithm on multimodal problems by detecting promising areas," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1438–1449, Aug. 2015.
[58] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," School EEE, Nanyang Technol. Univ., Singapore, KanGAL Rep. 2005005, 2005.
[59] 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, Sep. 2009.
[60] X. Jin, Y. Q. Liang, D. P. Tian, and F. Z. Zhuang, "Particle swarm optimization using dimension selection methods," *Appl. Math. Comput.*, vol. 219, no. 10, pp. 5185–5197, Jan. 2013.
[61] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Inf. Sci.*, vol. 293, no. 3, pp. 370–382, Feb. 2015.
[62] D. Swagatam and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC2011 competition on testing evolutionary algorithm on real world optimization problem," Dept. Electron. Telecommun. Eng., Jadavpur University, Kolkata, India, Rep., Dec. 2010.
[63] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

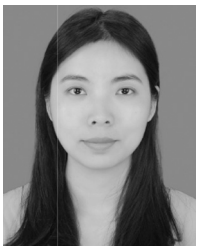
Xuewen Xia (M'19) received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, China, in 2009.

In 2009, he was a Lecturer with Hubei Engineering University, Xiaogan, China. In 2012, he was a Postdoctoral Researcher with Wuhan University. In 2014, he was an Associate Professor with the School of Software, East China Jiaotong University, Nanchang, China. He is currently a Professor with the College of Physics and Information Engineering, Minnan Normal University, Zhangzhou, China. His current research interest includes computational intelligence techniques and their applications.

Ling Gui, photograph and biography not available at the time of publication.

Fei Yu received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, China, in 2015.

He is currently an Associate Professor with Minnan Normal University, Zhangzhou, China. His current research interests include intelligent computing and machine learning.



Hongrun Wu received the Ph.D. degree from Wuhan University, Wuhan, China, in 2018.

She is currently a Lecturer with the School of Physics and Information Engineering, Minnan Normal University, Zhangzhou, China. Her current research interests include computational intelligence and its applications in the field of complex networks, graph neural networks, and computer vision.

Bo Wei received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, China, in 2013.

He is currently a Lecturer with East China Jiaotong University, Nanchang, China. His current research interests include intelligent computation and machine learning.

Ying-Long Zhang received the Ph.D. degree from the Renmin University of China, Beijing, China, in 2014.

He is an Assistant Professor with Minnan Normal University, Zhangzhou, China. His current research interests include data mining and information network analysis.

Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Prof. Zhan was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018 and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. His doctoral dissertation was awarded the China Computer Federation Outstanding Ph.D. dissertation and the IEEE Computational Intelligence Society Outstanding Ph.D. dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of *Neurocomputing*.