An Improved Method for Comprehensive Learning Particle Swarm Optimization

Zi-Jia Wang, *Student Member, IEEE*, Zhi-Hui Zhan (Corresponding Author), *Member, IEEE*, and Jun Zhang, *Senior Member, IEEE*

***Abstract*—Particle swarm optimization (PSO) is a population-based stochastic search technique for solving optimization problems, which has been proven to be effective in wide applications in scientific and engineering domains. However, standard PSO is inefficient when searching in complex problems spaces. Lots of improved PSO variants with different features have been proposed, such as comprehensive learning PSO (CLPSO). CLPSO is an enhanced PSO variant by adopting a better learning strategy that lets particle have some chance to choose other particles' historically best information to update the velocity. Comparing with the standard PSO, CLPSO has successfully improved the diversity of population and hence avoids the deficiency of premature convergence and local optima. However, CLPSO causes slow convergence speed, especially during the late state of searching process. In this paper, an improved CLPSO algorithm is proposed, termed as ICLPSO, to accelerate convergence speed and keep diversity of population at the same time. We set the learning probability based on particles' own fitness and adaptively construct different learning exemplars for different particles according to particles' own features and properties. This is a more appropriate learning strategy for particles' optimization, rather than the random selection fashion in CLPSO. Experimental results show that the performance of ICLPSO is better than standard CLPSO and some other peer algorithms, in terms of both unimodal and multimodal functions.**

I. INTRODUCTION

PARTICLE swarm optimization (PSO), [1], [2], which was first introduced by Kennedy and Eberhart in 1995, is one of the most important swarm intelligence algorithms [3]. It emulates swarm behaviors such as birds flocking and fish schooling to search for an optimum. It is initialized with a population of particles randomly distributed in the search space. Each particle in population is associated with two vectors: a velocity vector and a position vector. During the process, each particle flies in the search space and adjusts its flying velocity according to its personal best experience (the so-called *pbest*) and the position of the known best-fit particle in the entire population (the so-called *gbest*). As PSO is easy to accomplish, it has rapidly attracted attention from many scientists and applied to real-world optimization problems [4]-[12].

As many real-world optimization problems become increasingly complex, better optimization algorithms are always needed. However, the standard PSO algorithm may easily get trapped in local optima when solving some complex multimodal problems [13]. Besides, PSO is also a population-based iterative algorithm. Therefore, it may be computationally inefficient as measured by the number of function evaluations (FEs) required [14]. These weaknesses have restricted wider applications of the PSO.

Thus, accelerating convergence speed and avoiding local optima have become the two important goals [15]. Motivated by these goals, a number of PSO variants have been proposed to overcome these problems [15]-[22]. For example, Liang et al. [16] introduced a comprehensive learning particle swarm optimizer (CLPSO). However, so far, it seems difficult to simultaneously achieve both goals. CLPSO can avoid local optima, but also slows the convergence speed, especially during the late state of searching process.

Motivated by this problem, in this paper, we proposed an improved CLPSO algorithm (ICLPSO), which is a variant of CLPSO. ICLPSO generates the learning probability *Pc* based on particles' own fitness and adaptively constructs different learning exemplars for different particles according to particles' own features and properties, which is a more suitable learning strategy for particles' optimization. It can help the better particles keep their properties and help the worse particles have more chance to learn other particles with better fitness. Experimental results show that the performance of ICLPSO is more effective and more accurate than standard CLPSO and some other peer algorithms in dealing with optimization functions no matter unimodal or multimodal. It can accelerate convergence speed and keep population diversity at the same time.

The rest of the paper is organized as follows. Section II introduces the standard PSO and its developments briefly.

Z.-J. Wang, Z.-H. Zhan, and J. Zhang are with the Department of Computer Science, Sun Yat-Sen University, Guangzhou, 510275, China, with the Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China, with the Engineering Research Center of Supercomputing Engineering Software (Sun Yat-sen University), Ministry of Education, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, China. Z.-H. Zhan is the corresponding author of this paper, email: zhanzhh@mail.sysu.edu.cn.

This work was partially supported by the National Natural Science Foundations of China (NSFC) with No. 61402545, the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the NSFC Key Program with No. 61332002, the NSFC for Distinguished Young Scholars with No. 61125205, the Fundamental Research Funds for the Central Universities (15lgzd08), and the National High-Technology Research and Development Program (863 Program) of China No.2013AA01A212.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.41

IEEE computer society

ICLPSO based on CLPSO is proposed in Section III. Section IV compares ICLPSO with CLPSO and other algorithms using a set of benchmark functions and makes discussions. Conclusions are given in Section V.

II. PSO AND SOME VARIANTS

A. PSO Framework

In PSO, the member of the swarm is called particle, which means a possible solution in the search space. Each particle *i* is associated with two vectors. The vector $V_i = [v_i^1, v_i^2, ..., v_i^D]$ means the velocity of the particle *i*, while the vector $X_i = [x_i^1, x_i^2, ..., x_i^D]$ means the position of the particle *i*, where *D* stands for the dimensions of solution space. Both the two vectors are initialized randomly within the corresponding ranges. At the end of each iteration step, each particle in population adjusts its position based on its own *pbest* and the *gbest* in the entire population. The position *X* and velocity *V* of each particle are updated according to the following formula:

$$v_i^d = \omega * v_i^d + c_1 * rand1_i^d * (pbest_i^d - x_i^d) + c_2 * rand2_i^d * (gbest^d - x_i^d) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

where ω is the inertia weight [17], to balance the global and local search performance. c_1 and c_2 are the acceleration coefficients, often set as 2.0. Parameter c_1 pulls the particle to its own *pbest*, ensuring the diversity of population; while c_2 pushes the swarm to converge to the current *gbest*, ensuring the speed of convergence. $rand1_i^d$ and $rand2_i^d$ are two uniformly distributed random numbers within [0, 1]. A particle's velocity and position on each dimension are clamped to the maximum $V_{\max}$ and $X_{\max}$.

The flowchart of PSO is shown in Fig. 1.

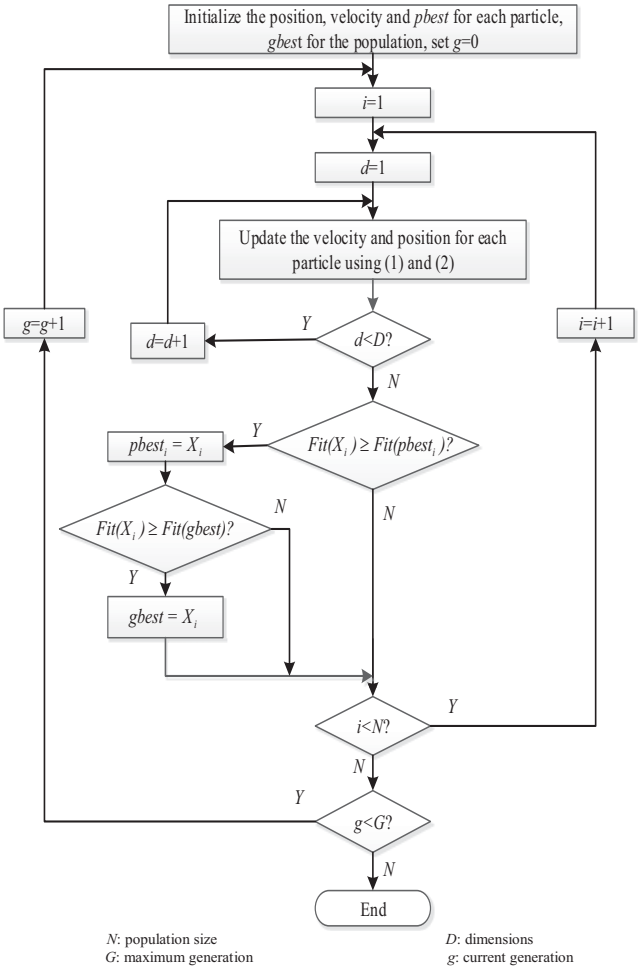

Fig. 1. Flowchart of the PSO framework

B. Current Developments of the PSO

Since its introduction in 1995 by Kennedy and Eberhart, [1], [2], PSO has become a popular optimizer and has been applied in problem solving due to its simple concept and effectiveness. In the literature, many researches have worked on improving its performance in various ways.

Shi and Eberhart [17] proposed the large inertia weight ω is appropriate for global search, while the small inertia weight ω is suitable for local search. In that way, ω should decrease linearly during the evolution process. It can be described as:

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * \frac{g}{G} \quad (3)$$

where *g* is the current evolutionary generation, and *G* is the maximum number of generation. Besides, the parameter $\omega_{\max}$ and $\omega_{\min}$ are often set as 0.9 and 0.4.

In addition, improving PSO's performance by designing different types of topologies is also an effective method. Generally, there are two major variants of PSO algorithms depending on the topology [18]. The *gbest* model (GPSO) shares information among the whole swarm and every particle is able to obtain information from the best particle in the entire swarm population. The *lbest* model (LPSO), making communication and utilizing information within a small group of particles. Mendes and Kennedy [19] introduced a fully informed particle swarm (FIPS). In the fully informed neighborhood, all neighbors are a source of influence. There are five different PSO topologies often used called all, ring, four clusters, pyramid and square. The ring topology is a well-known local topology. In the ring topology, each particle connects with only two of other particles in the swarm and only makes use of the information from the two particles while updating its own position.

Comprehensive learning particle swarm optimizer (CLPSO) was first introduced by Liang et al. [16]. In CLPSO, particles make use of different particles' *pbest* values to update the velocity of different dimensions as:

$$v_i^d = \omega * v_i^d + c * rand_i^d * (pbest_{fi(d)}^d - x_i^d) \quad (4)$$

where ω is the inertia weight as in (1), c is the acceleration coefficient fixed to be 1.49445, and $rand_i^d$ is a random value within [0, 1]. The *fi(d)* is the particle's index which used to guide the flying in the dth dimension of particle i, which can be any particle including the particle i itself.

In order to find the *fi(d)* for each dimension, CLPSO firstly generates a random number r, then compares r with Pc_i, which is the learning probability to control particles' learning from self or others, can be seen in (5).

$$Pc_i = 0.05 + 0.45 * \frac{(\exp(\frac{10(i-1)}{ps-1}) - 1)}{(\exp(10) - 1)} \quad i=1,2,\ldots\ldots ps \qquad (5)$$

where ps is the population size. If r is smaller than Pc_i, then this dimension learns from others, otherwise learns from itself. When learning from others, CLPSO chooses two particles excluding the particle whose velocity is updating. And then compares the fitness values of these two particles' *pbest*s and selects the one with better fitness as exemplar. If all exemplars come from the particle's own *pbest*, CLPSO randomly chooses a dimension to learn from another particle's *pbest*'s corresponding dimension. A particle will keep learning from its exemplar until it cannot improve the solution quality for several generations which is called refreshing gap m, then the new learning exemplar will be chosen again using the same method. The refreshing gap in CLPSO is set as 7 based on experiments.

And the details of choosing learning exemplar in CLPSO are given in Algorithm 1. (We define the fitness value the larger the better). Fig. 2 presents an example of *Pc* for a population size of 20.

Besides, CLPSO also uses a different method to constrain the particles within the range as follows: Calculate the fitness value of a particle and update its *pbest* only if the particle is in the range. Since all exemplars are within the range, the particle will eventually return to the search range.

Algorithm1 generating learning exemplar in CLPSO

```
d=1
while d≤D
    if r<Pc_i
       Randomly choose two particles out of the population which
            excludes the particle whose velocity is updating
       Compare the fitness values of the two particles' pbests
       The d^th dimension will learn from the better one's pbest
    else
       The d^th dimension will learn from its own pbest
    end if
    d=d+1
end while

r: random number
```

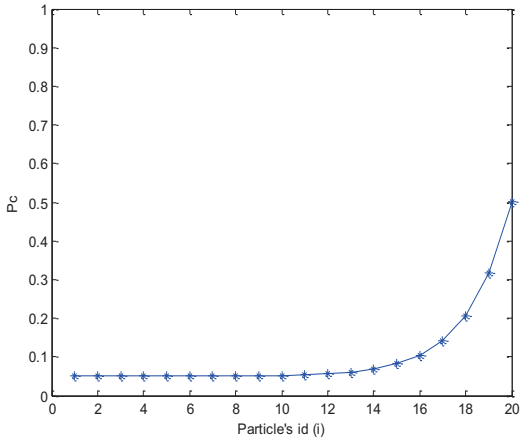

Fig. 2. *Pc* for a population size of 20 in CLPSO

III. ICLPSO APPROACH

As we mentioned above, CLPSO generates each learning probability Pc_i using formula (5). The probabilities are determined by particles' indexes and will remain unchanged throughout the running process. However, in a typical run of PSO, the features and properties of particles are changed along with their evolution. As a result, it would make the algorithm more effective if the learning probability can change accordingly.

Besides, the method of constructing learning exemplar (given in Algorithm 1) is invariant of the particles' fitness. Noticing that different particles have different features, it would make algorithm more intelligent if the process can be adjusted adaptively. Motivated by these findings, we proposed an improved CLPSO, which is detailed in the following subsections.

A. Generate the Learning Probability Pc for Each Particle

We set Pc_i according to the i^{th} particle's fitness. Because generating the Pc_i using formula (5) is only related to particles' indexes and will remain unchanged, however, the index makes nearly no sense on particles' optimization. Besides, the performance of each particle is changed at each iteration step, so the learning probability Pc_i should also be changed dynamically. So, our method is, when the fitness is large, which means the particle is good, we make the probability of learning other particle's *pbest* smaller and make the probability of learning its own *pbest* larger. When the fitness is low, in other words, the particle is relatively poor, we make the probability of learning other particle's *pbest* larger and make the probability of learning its own *pbest* smaller. (We define the fitness value the larger the better). Our method of calculating the learning probabilities is described as follows:

(1) Sorting particles in descending order according to their own fitness at the beginning of each iteration step;

(2) $Pc_i = i^{'}/(2*N)$ (6)

where N is the population size, i' is the position of particle i^{th} in the sort of fitness.

In this way, if i' is small, which means the particle i has a good fitness value, we make Pc_i smaller in order to make this particle learn from itself with a higher probability to keep its own property. On the contrary, if i' is large, which means the particle i has a relatively bad fitness value, we make Pc_i larger in order to make particle i has a larger probability to learn from other particles. The range of Pc_i in expression (6) is [1/ (2*N), 1/2].

B. Construct the Learning Exemplar

In CLPSO, it only chooses two particles to compare, so the probability of choosing top-ranking particles is small. Although it can enhance the diversity of population, it slows the convergence speed. Besides, different particles have different properties, so this method should also be adjusted adaptively. In our method, we randomly choose n particles out of the population which excludes the particle whose velocity is updating. Then, compare the fitness values of these particles' *pbest* and select the best one. The definition of parameter n is also related to particle's fitness, similar to that of learning probability:

(1) Sorting particles in descending order according to their own fitness at the beginning of each iteration step;

(2) n=2+round(([N/2]-2)/N*i') (7)

where N is the population size, i' is the position of particle i^{th} in the sort of fitness, ⌈ ⌉ is ceiling operator.

That is, if i' is small, the fitness of particle is large, which means the particle is good and many particles in population are worse than it. At this time, there is no need to choose many particles, choosing fewer particles can keep the diversity of population. So we make n smaller. If i' is large, the fitness of particle is low, the particle is relatively poor, many particles in population are better than it. As a result, we make n larger. So we can choose more particles from population to compare in order to have a larger probability of choosing better particles. In that way, we can accelerate the speed of convergence. However, we can't choose the best particle from all the population, similar to the traditional PSO, which may get trapped in the local optimal and cause premature convergence. So the range of n in expression (7) is [2, ⌈N/2⌉].

There are three main advantages of ICLPSO as follows:

(1) Using the method similar to CLPSO to construct learning exemplar, keep the better search ability and diversity of population.

(2) Changing the learning probability Pc_i dynamically depends on particle's own fitness can help better particles keep their own features and properties and help the worse particles have more chance to learn other particle with better fitness.

(3) Choosing n (n∈[2, ⌈N/2⌉]) particles out of the population to compare has the larger probability to learn better particles, inhibit the randomness of the algorithm and accelerate the convergence speed. The details of choosing learning exemplar in ICLPSO are given in Algorithm 2.

Algorithm2 generating learning exemplar in ICLPSO

```
Generate parameter n using formula (7)
d=1
while d≤D
    if r<Pc_i
        Randomly choose n particles out of the population which
            excludes the particle whose velocity is updating
        Compare the fitness values of these n particles' pbests
        The d^th dimension will learn from the best one's pbest
    else
        The d^th dimension will learn from its own pbest
    end if
    d=d+1
end while

r: random number
```

In other words, ICLPSO combines the advantages between PSO and CLPSO. And with these three advantages, ICLPSO can easily get optimal solutions and maintain them until the end of the predefined budget of the function evaluations for unimodal or multimodal optimization. The details of ICLPSO algorithm are shown in Fig. 3.

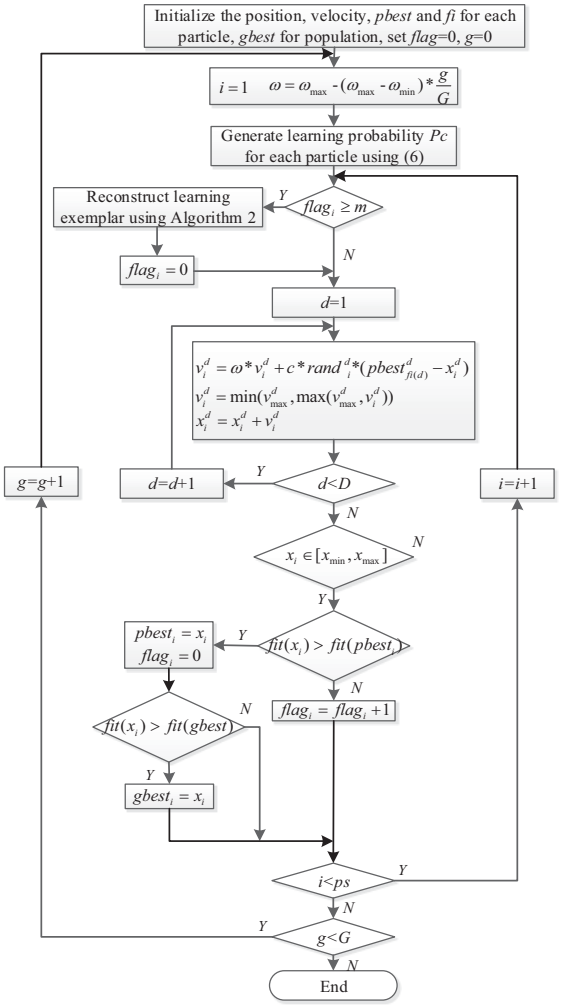

Fig. 3. Flowchart of the ICLPSO algorithm

IV. Experiment Testing And Results

A. Experimental Settings

In this section, ICLPSO is applied to minimize a set of 13 benchmark functions of dimensions D = 30 chosen from [23] as shown in Table I. The first seven functions f_1–f_7 are unimodal functions. These functions are used to test the convergence speed of these algorithms. The next six functions f_8–f_{13} are multimodal functions where the number of local optima increases exponentially with the problem dimension.

The population size NP of ICLPSO is set to 30 for each problem. We also set the refreshing gap m and acceleration

TABLE I. THIRTEEN HIGH-DIMENSIONAL TEST FUNCTIONS USED IN THIS PAPER, BOTH UNIMODAL AND MULTIMODAL

Name	Test functions	Initial range	$f_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0
Schewefel 2.22	$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0
Schewefel 1.2	$f_3(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-100,100]^D$	0
Schewefel 2.21	$f_4=\max_i\{\lvert x_i\rvert\}$	$[-100,100]^D$	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$[-30,30]^D$	0
Step	$f_6(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	$[-100,100]^D$	0
Noisy Quaric	$f_7(x)=\sum_{i=1}^{D}ix_i^4+rand[0,1)$	$[-1.28,1.28]^D$	0
Schewefel 2.26	$f_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^D$	-12569.5 for D=30
Rastrigin	$f_9(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^D$	0
Ackley	$f_{10}=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	$[-32,32]^D$	0
Griewank	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^D$	0
Penalized	$f_{12}(x)=\frac{\pi}{D}\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\}$ $+\sum_{i=1}^{D}u(x_i,10,100,4)$, where $y_i=1+\frac{1}{4}(x_i+1)$	$[-50,50]^D$	0
	$f_{13}=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^{D}u(x_i,5,100,4)$ where $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	$[-50,50]^D$	0

TABLE II. EXPERIMENTAL RESULTS OF 30-DIMENSIONAL PROBLEMS f_1-f_{13}, AVERAGED OVER 30 INDEPENDENT RUNS

F	FEs	ICLPSO		CLPSO		FIPS		PSO	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	300000	4.16E-47	3.78E-47	1.72E-21(-)	2.38E-21	7.18E-29(-)	1.33E-29	**3.56E-53**(+)	7.33E-53
f_2	300000	1.16E-32	3.41E-32	1.44E-13(-)	2.05E-13	5.17E-18(-)	2.38E-18	**5.28E-34**(+)	6.37E-34
f_3	300000	7.10E-01	3.85E-01	1.29E+02(-)	5.44E+01	1.47E+00(-)	9.15E-01	**1.15E-01**(+)	1.32E-01
f_4	300000	5.74E-02	9.45E-02	1.36E+00(-)	7.85E-01	**6.35E-05(+)**	3.37E-05	4.05E-01(-)	2.58E-01
f_5	300000	**1.21E+01**	7.11E+00	2.47E+01(-)	1.72E+01	2.14E+01(-)	9.38E+00	2.38E+01(-)	1.25E+01
f_6	300000	**0.00E+00**	0.00E+00	**0.00E+00(≈)**	0.00E+00	**0.00E+00(≈)**	0.00E+00	**0.00E+00(≈)**	0.00E+00
f_7	300000	**1.11E-03**	2.83E-03	3.56E-03**(-)**	4.12E-03	4.35E-03(-)	5.51E-03	3.58E-03(-)	5.24E-03
f_8	300000	**-12569.5**	5.19E-12	**-12569.5(≈)**	3.81E-12	-6771.4(-)	6.35E+02	-10488.7(-)	5.36E+02
f_9	300000	**2.84E-15**	1.59E-15	5.24E-10(-)	3.92E-10	3.22E+01(-)	1.37E+01	2.96E+01(-)	9.81E+00
f_{10}	300000	2.58E-14	2.63E-14	1.43E-12(-)	2.48E-13	**7.98E-15**(+)	6.61E-15	3.56E-14(≈)	4.38E-14
f_{11}	300000	**0.00E+00**	0.00E+00	4.77E-13(-)	6.11E-13	**0.00E+00(≈)**	0.00E+00	2.18E-02(-)	7.33E-02
f_{12}	300000	**1.57E-32**	2.78E-48	4.89E-22(-)	1.46E-21	5.25E-29(-)	3.81E-29	7.91E-02(-)	6.27E-02
f_{13}	300000	**1.35E-32**	5.57E-48	6.21E-21(-)	9.19E-22	6.77E-30(-)	4.98E-30	5.11E-04(-)	3.22E-03
+(better than ICLPSO)				0		2		3	
-(worse than ICLPSO)				11		9		8	
≈(no sig.)				2		2		2	

≈, +, - indicates whether a given algorithm performed no significantly different, better or worse compared to ICLPSO according the Wilcoxon rank-sum test

coefficient c in our algorithm is 7 and 1.49445. Besides, the method to constrain particles within the range is same as CLPSO. According to [19], the parameter χ and $\varphi_{\max}$ in FIPS is set as 0.7298 and 4.1, and we use a ring topology in our experiment. Each experiment is run 30 times independently and the results are averaged. For clarity, the results of the best algorithms are marked in **boldface**, respectively. In addition, we make use of the Wilcoxon's rank sum test [24] at $\alpha = 0.05$ to evaluate the statistical significance of the results. The ≈, +, - indicate whether a given algorithm performed no significantly different (≈), better (+) or worse (-) compared to ICLPSO according the Wilcoxon rank-sum test.

B. Results and Discussions

- We can see the statistical results in Table II. As for unimodal functions, PSO has fast convergence speed, so it has a relatively better performance, especially on f_1-f_3. While the convergence speed in CLPSO is slow. But our method ICLPSO well solves the problem of slow convergence speed from CLPSO, the error from ICLPSO is smaller than CLPSO and the result is relatively stable in ICLPSO. Although it is a little worse than PSO, it is better than the other two algorithms obviously.
- As for the step function in f_6, which is also unimodal, these four algorithms can all get the optimal solution.
- The functions f_8-f_{13}, which are multimodal, ICLPSO keeps the diversity of population, and accelerate the convergence speed at the same time. The error from ICLPSO is also smaller than CLPSO and the result is relatively stable in ICLPSO, especially in f_8 and f_{11}, ICLPSO can get the optimal solution. Comparatively speaking, PSO may trap in local optima and cause premature convergence, it has a relatively worse performance on these functions.

All in all, as shown in Table II, compare with CLPSO, ICLPSO performs better on 11 functions while the results of the other functions are not significantly different. Besides, ICLPSO obtains better results on 9 functions than FIPS, only worse on f_4 and f_{10}. For standard PSO, ICLPSO performs better on 8 functions and only worse on the first 3 functions due to the fast convergence speed during the late state in PSO. So, we can conclude that ICLPSO outperforms CLPSO, FIPS, and PSO.

Table III summarizes the success rate (SR) of each algorithm which acceptable solutions are found over 30 runs and the average number of function evaluations (FEs) required to find the acceptable solutions. FEs and SR are useful to compare the convergence rate and stability of different algorithms. The ranks in table are evaluated based on the descending order of the success rates and the ascending order of FEs. From Table III, we find only ICLPSO 100% successfully finds the acceptable solutions on all functions in particular. And ICLPSO ranks the first and is the fastest in most of the functions. Overall, ICLPSO performs the best on most functions in f_1-f_{13}. The results show that the new method of generating learning probability and constructing learning exemplar speeds up the search process, and move to optima quickly. Next, we will see the convergence of these four algorithms more apparently using the method of scatter diagram. We choose unimodal functions f_1, f_2 and multimodal functions f_8, f_{13} as examples to illustrate it.

TABLE III. SUCCESS RATE AND SEARCH SPEED COMPARISONS ON 30-DIMENSIONAL PROBLEMS F_1-F_{13}

fun	Acceptable accuracy		ICLPSO	CLPSO	FIPS	PSO
f_1	1E-10	SR	**100**	**100**	**100**	**100**
		FEs	**1.09E+05**	1.86E+05	1.55E+05	2.00E+05
		Rank	**1**	3	2	4
f_2	1E-10	SR	**100**	**100**	**100**	**100**
		FEs	**1.36E+05**	2.49E+05	2.13E+05	2.10E+05
		Rank	**1**	4	3	2
f_3	1E+00	SR	**100**	0	43.3	**100**
		FEs	2.74E+05	N/A	2.79E+05	**2.66E+05**
		Rank	2	4	3	**1**
f_4	1E+00	SR	**100**	60	**100**	**100**
		FEs	1.88E+05	2.46E+05	**1.29E+05**	2.43E+05
		Rank	2	4	**1**	3
f_5	1E+02	SR	**100**	**100**	**100**	**100**
		FEs	**5.87E+04**	1.18E+05	8.32E+04	1.67E+05
		Rank	**1**	3	2	4
f_6	0	SR	**100**	**100**	**100**	**100**
		FEs	**5.42E+04**	8.09E+04	5.88E+04	1.63E+05
		Rank	**1**	3	2	4
f_7	1E-2	SR	**100**	**100**	**100**	**100**
		FEs	**1.05E+05**	1.38E+05	1.82E+05	1.71E+05
		Rank	**1**	2	4	3
f_8	-10000	SR	**100**	**100**	0	56.7
		FEs	**3.54E+04**	4.77E+04	N/A	2.30E+05
		Rank	**1**	2	4	3
f_9	1E-5	SR	**100**	**100**	0	0
		FEs	**2.27E+05**	2.69E+05	N/A	N/A
		Rank	**1**	2	3.5	3.5
f_{10}	1E-10	SR	**100**	**100**	**100**	**100**
		FEs	**1.56E+05**	2.77E+05	2.04E+05	2.56E+05
		Rank	**1**	4	2	3
f_{11}	1E-10	SR	**100**	**100**	**100**	13.3
		FEs	**1.13E+05**	2.51E+05	1.71E+05	2.04E+05
		Rank	**1**	3	2	4
f_{12}	1E-10	SR	**100**	**100**	**100**	76.7
		FEs	**1.12E+05**	2.23E+05	1.44E+05	2.48E+05
		Rank	**1**	3	2	4
f_{13}	1E-10	SR	**100**	**100**	**100**	83.3
		FEs	**1.33E+05**	2.14E+05	1.66E+05	2.46E+05
		Rank	**1**	3	2	4
Avg_Rank			**1.15**	3.08	2.5	3.27

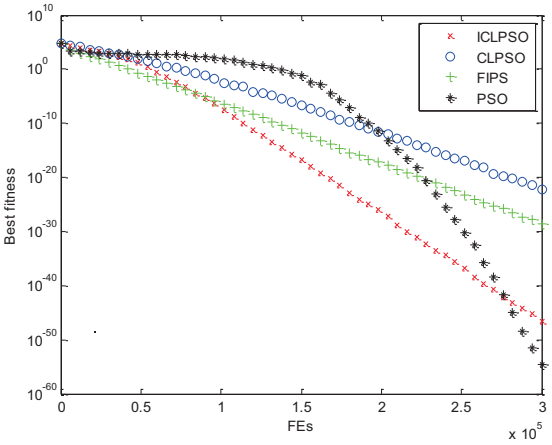

Fig. 4. The results from four algorithms on f_1, FEs=300000

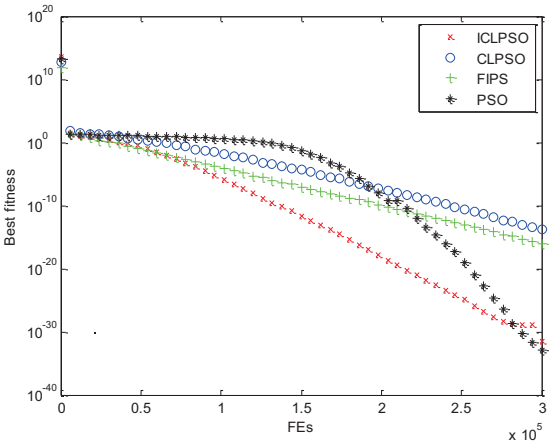

Fig. 5. The results from four algorithms on f_2, FEs=300000

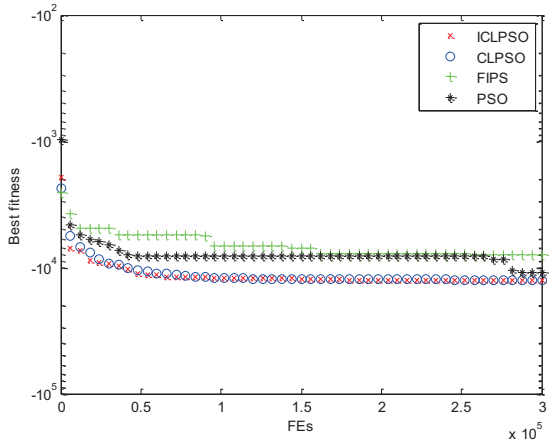

Fig. 6. The results from four algorithms on f_8, FEs=300000

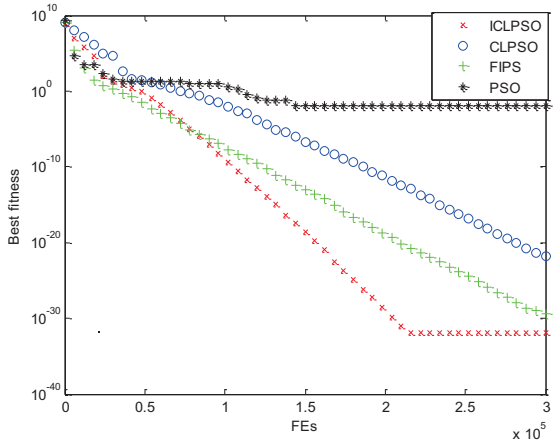

Fig. 7. The results from four algorithms on f_{13}, FEs=300000

In order to show the convergence characteristics of these compared algorithms, we plot the average evolutionary curve during the running time of some selected functions in Fig. 4, Fig. 5, Fig. 6, and Fig. 7. From the figures presented above, we can see that:

- For unimodal functions, PSO has fast convergence speed, especially during the late state of evolutionary process. ICLPSO has a little slower convergence speed than PSO, however, it performs much better than the other two algorithms.
- For multimodal functions, PSO may get trapped in some local optima, while the other three algorithms can keep the diversity to some degree. But FIPS performs worse in f_8. We can also see that the error in ICLPSO is smaller than CLPSO and FIPS, the convergence speed is also faster than these two algorithms. In other words, ICLPSO well solves the problem of slow convergence speed from CLPSO.

The better performance is due to the better choosing learning probability *Pc* depends on fitness rather than particle's own index. Of course, changing the method of generating learning exemplar adaptively is also more suitable for optimization. As ICLPSO combines the advantages of PSO and CLPSO, the search can be easily located at optimal solutions, which is suitable for both unimodal and multimodal functions.

V. Conclusion

In this paper, an improved CLPSO algorithm (ICLPSO) was proposed to achieve optimization. ICLPSO made use of the benefits both from the particle swarm optimizer (PSO) and comprehensive learning particle swarm optimizer (CLPSO). With the better method of choosing learning probability *Pc* and generating learning exemplar, ICLPSO is more suitable for optimization. Experimental results also showed that the proposed ICLPSO algorithm can perform better than CLPSO

and some other peer algorithms in this paper on a suite of widely used test functions both unimodal and multimodal in a statistically meaningful way.

In the future, the proposed ICLPSO is expected to be applied on more complex optimization problems like topological active net optimization [25], cloud computing resources scheduling [26][27], and even in a dynamic environment [28].

REFERENCES

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

[2] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci.*, Nagoya, Japan, 1995, vol. 1, pp. 39–43.

[3] J. Kennedy, R. C. Eberhart, and Y. H. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[4] Z. H. Zhan, J. Li, J. Cao, J. Zhang, H, Chung, and Y. H. Shi, "Multiple populations for multiple objectives: a coevolutionary technique for solving multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 445-463, 2013.

[5] S. Mandal, P. Mallick, D. Mandal, et al. "Optimal FIR band pass filter design using novel particle swarm optimization algorithm," *IEEE Symposium on Humanities, Science and Engineering Research (SHUSER)*, 2012, pp. 141-146.

[6] Y. J. Gong, J. Zhang, "Real-time traffic signal control for roundabouts by using a PSO-based fuzzy controller", *IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1-8.

[7] S. Kibria, M. T. Islam, B. Yatim, "New compact dual-band circularly polarized universal RFID reader antenna using ramped convergence particle swarm optimization," *IEEE Trans. Antennas and Propagation*, 2014, vol. 62, no. 5, pp. 2795–2801.

[8] W. Hu, and G. G. Yen, "Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, 2015, pp. 1-18.

[9] A. Deb, J. S. Roy, and B. Gupta, "Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas." *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 8, 2014, pp. 3920-3928.

[10] M. Shen, Z. H. Zhan, W. N. Chen, Y. J. Gong, J. Zhang and Y. Li, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, 2014, pp. 7141-7151.

[11] 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," *Information Sciences*, vol. 293, no. 1, pp. 370-382, 2015.

[12] H. B. Duan, P. Li, and Y. X. Yu, "A predator-prey particle swarm optimization approach to multiple UCAV air combat modeled by dynamic game theory," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 1, 2015, pp. 11-18.

[13] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," in *Information Processing Letters*, 2003, vol. 85, no .6, pp. 317-325.

[14] G. Ciuprina, D. Ioan, I. Munteanu, "Use of intelligent-particle swarm optimization in electromagnetics," *IEEE Transactions on Magnetics*, 2002, vol. 38, no. 2, pp. 1037-1040.

[15] Z. H. Zhan, J. Zhang, Y. Li, and H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 2, pp. 1362-1381, 2009.

[16] 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.*, 2006, vol. 10, no. 3, pp. 281–295.

[17] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69–73.

[18] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in Proc. *IEEE Swarm Intelligence Symposium*, 2007, pp.120–127.

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

[20] X. D. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Trans. Evol. Computat.*, 2010, vol. 14, no. 1, pp. 150-169.

[21] B. Y. Qu, P. N. Suganthan, and S. Das. "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Trans. Evol. Computat.*, 2013, vol. 17, no. 3, pp. 387-402.

[22] X. D. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genetic and Evolutionary Computation Conference*, Jun. 2004, pp. 105-116.

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

[24] J. Derrac, S. Garc´ıa, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[25] Y. L. Li, Z. H. Zhan, Y. J. Gong, J. Zhang, Y. Li, and Q. Li, "Fast micro-differential evolution for topological active net optimization," *IEEE Transactions on Cybernetics*, DOI:10.1109/TCYB.2015.2437282, 2015.

[26] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, H. S. H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol. 47, no. 4, Article 63, pp. 1-33, Jul. 2015.

[27] H. H. Li, Z. G. Chen, Z. H. Zhan, K. J. Du, and J. Zhang, "Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment," in *Proc. Genetic Evol. Comput. Conf.*, Madrid, Spain, Jul. 2015, pp. 1419-1420.

[28] Z. H. Zhan, J. Zhang, and J. J. Li, "Adaptive particle swarm optimization with variable relocation for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, Jul. 2014, pp. 1565-1570.