Information Sciences 436–437 (2018) 162–177

Contents lists available at ScienceDirect

Information Sciences

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

A hybrid particle swarm optimization algorithm using adaptive learning strategy

Feng Wang[a,*], Heng Zhang[a], Kangshun Li[b], Zhiyi Lin[c], Jun Yang[d], Xiao-Liang Shen[e]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[c] School of Computer Science, Guangdong University of Technology, Guangzhou 510006, China
[d] School of Electrical Engineering, Wuhan University, Wuhan 430072, China
[e] School of Economics and Management, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Article history:
Received 5 November 2017
Revised 2 January 2018
Accepted 13 January 2018
Available online 31 January 2018

Keywords:
Particle swarm optimization
Learning strategy
Search direction
Multimodal optimization

ABSTRACT

Many optimization problems in reality have become more and more complex, which promote the research on the improvement of different optimization algorithms. The particle swarm optimization (PSO) algorithm has been proved to be an effective tool to solve various kinds of optimization problems. However, for the basic PSO, the updating strategy is mainly aims to learn the global best, and it often suffers premature convergence as well as performs poorly on many complex optimization problems, especially for multimodal problems. A hybrid PSO algorithm which employs an adaptive learning strategy (ALPSO) is developed in this paper. In ALPSO, we employ a self-learning based candidate generation strategy to ensure the exploration ability, and a competitive learning based prediction strategy to guarantee exploitation of the algorithm. To balance the exploration ability and the exploitation ability well, we design a tolerance based search direction adjustment mechanism. The experimental results on 40 benchmark test functions demonstrate that, compared with five representative PSO algorithms, ALPSO performs much better than the others in more cases, on both convergence accuracy and convergence speed.

1. Introduction

Many real-world problems which can be transferred to the optimization problems now have become more and more complex, and are difficult to be solved by the canonical optimization algorithms. Therefore, the research on optimization algorithms in real applications has become a promising research issue. In the past few years, lots of meta-heuristic optimization algorithms have emerged [4,15,24,25,29,30]. And one of the most popular algorithm is particle swarm optimization (PSO).

PSO is a nature-inspired evolutionary algorithm firstly proposed in 1995 by Kennedy et al. [6]. In PSO, each solution is represented by a particle in the population with two main vectors, namely velocity and position, which are dynamically changed according to its interactions with other particles during the evolutionary process. And the particles can adjust their trajectories by their flying experiences in each generation which can help them fly towards a better search space. PSO can

* Corresponding author.
E-mail address: fengwang@whu.edu.cn (F. Wang).

https://doi.org/10.1016/j.ins.2018.01.027

be implemented easily to deal with many function optimizations with fast convergence, and in the last few decades, PSO has been widely applied in many real optimization problems and has shown good performances, such as manufacturing control in engineering optimization [5,16,17,22,27], and multi-source scheduling in cloud computing [9–11].

However, the basic PSO also has some drawbacks which make it get trapped in the local optimum and suffer the premature convergence, especially in the large scale complex optimization problems. Firstly, for basic PSO, the particles always adjust their flying trajectories and converge to a single point regarding the parameter settings, and different parameter settings may lead to different convergence speed. Secondly, the particles evolve according to their interactions with each other during the search process, and the updating strategy is mainly aims to the global best particle which makes the population lose the diversity that increases the possibilities of getting trapped in local optima.

Hence, how to improve the convergence speed as well as avert the premature convergence has become the most important research problem in PSO. In recent years, lots of research works have been done and many variants of PSO have been put forward. These algorithms try to balance the local search ability and global search ability through the tuning of the parameters, modification of the updating rules, designing new strategies in the evolving process and indeed get some improvements on the convergence of the solutions. Nevertheless, with the optimization problems becoming increasingly complex, the current algorithms still can't guarantee a good diversity and efficiency of the solutions in reality.

In this paper, in order to overcome the above limitations, a hybrid PSO algorithm with adaptive learning based strategy (ALPSO) is proposed, which incorporates a self-learning based candidate generation strategy to ensure exploration as well as a competitive learning based prediction strategy to guarantee exploitation of the algorithm, and a tolerance based search direction adjustment mechanism to well-balance exploration and exploitation.

The rest of this paper is organized as follows. A brief introduction of the standard PSO and its variants are presented in Section 2. The details of our proposed ALPSO algorithm is described in Section 3. The experimental results and analysis on several benchmark multimodal functions are shown in Section 4, and the conclusion is provided in Section 5.

2. PSO algorithm

2.1. Basic PSO

PSO is a swarm intelligence algorithm which was proposed based on the observation on the swarm behaviors in some ecological system, such as birds flying, bees foraging, or social behaviors of human beings. As one of the population-based algorithm, each particle in the population of PSO represents a possible solution of the optimization problem with two vectors (velocity and position). For a problem in D-dimensional space, a particle i which with the position $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ has a velocity $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ when the particle is moving. During the optimization process, the velocity vector and the position vector can be updated by the following rules:

$$\begin{aligned} v_{id}(t+1) &= w \cdot v_{id}(t) + c_1 \cdot r_1 \cdot (P_{id}(t) - X_{id}(t)) + c_2 \cdot r_2 \cdot (P_{gd}(t) - X_{id}(t)) \\ x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1) \end{aligned} \tag{1}$$

Where v_{id} denotes the velocity of the ith particle, x_{id} denotes the current position of the particle. And w denotes the inertia factor, r_1 and r_2 are two random numbers in the interval of [0, 1] which are employed to maintain the population's diversity, c_1 and c_2 are the accelerate coefficients which imply the relevant influence of the particles, $P_{id} = (pbest_{i1}, pbest_{i2}, \dots, pbest_{iD},)$ is the previous best position ($pbest$) of the ith particle and P_{gd} is the global best particle ($gbest$) found in the population.

2.2. Some variants of PSO

Due to the simplicity of implementation and efficiency of performance, PSO has become a promising tool for practical applications which has attracted many researchers to study and improve its performance. Recently, many variants of PSO algorithm have been proposed, which can be categorized into the following 3 types: 1) Parameter modification based PSO algorithms, which focus on the modification or adjustment methods of the inertia weights (w_i) and the accelerate coefficients (c_1 and c_2); 2) Population topology structure analysis based PSO algorithms, which try to use the information of population topology to guide the search process; 3) Evolutionary learning strategy based PSO algorithms, which integrate different evolutionary operators or use the historical information of the particles to design new strategies to improve the performance. Although many kinds of PSO variants have been proposed to avoid premature convergence, some strategies have been applied to keep the diversity of population and try to overcome the premature convergence, even some of them sacrifice the performance on convergence speed, it is still unavoidable for the population getting trapped in the local optima when dealing with difficult multimodal problems.

2.2.1. Parameter modification based PSO algorithms

As we mentioned above, the parameter setting has played a crucial role in PSO convergence behavior. Many research works have shown that, if the inertia weight becomes larger, the particles' speed will increase which will results in more exploration and less exploitation, and vice versa. And since the particles are updated by the cogitative component and social component, the control mechanism of these two coefficient parameters is also vital to the accuracy and efficiency of the solutions. How to find the appropriate values for the parameters is an challengeable research issue.

Shi et al. took the leading to employ an inertia weight in PSO which can loosen the restriction on velocity and control the search process better and proposed GPSO [20]. Afterwards, different kinds of adjustment strategy on w are advocated. Some researchers defined the inertia weight as a time-varing function [1] while some others took the adaptiveness into account and adjusted the value of inertia weight adaptively by monitoring the evolutionary states [21]. Ratnaweera et al. proposed a time-varying scheme for the acceleration coefficients, which help balance the global search ability and local search ability of the algorithm [18]. Zhan et al. use the information of the population distribution and particle fitness and propose an adaptive technique which can automatically control the inertia weight, the accreditation coefficients simultaneously[28].

2.2.2. Population topology structure analysis based PSO algorithms

In the standard PSO, the population size is fixed and each particle is evolved using the information got from all other particles. As we know that a larger population size may increase the computation costs resulting in slow convergence while a smaller population size may converge in local optimum. Followed by the idea of divided-and-conquer, many researchers focus on the topology structure and proposed many multi-swarm based PSO algorithms.

Different topology structures based on the connections of neighborhood have been employed to maintain the diversity of the solution, like LPSO [7]. Liang et al. proposed a CLPSO algorithm where every particle updates its velocity by learning from the historical best information of different particles which help the particles learn more valuable information to guide their search behaviors [13]. Then, they further investigated the sub-swarm method and proposed a DMS-PSO algorithm with dynamic multi swarms, in which the population is composed of many small sub-swarms and the dynamic strategy can keep the diversity of the population at the same time [14].

2.2.3. Evolutionary learning strategy based PSO algorithms

Since different optimization algorithms might have different strength, it is reasonable to learn or combine the strategies used in these algorithms to help to design a new one. Many evolutionary operators are introduced to increase the diversity of population and help PSO jump out of the local optimum. Some evolutionary algorithms, such as genetic algorithm (GA) [3], artificial bee colony algorithm (ABC) [26] , differential evolutionary algorithm (DE) [19] have been combined with PSO to improve the exploration ability.

Some other researchers investigated the flying behaviors of the particles and try to design proper learning strategies to deal with the information of the particles or swarms (sub-swarms), which are helpful for the solution jumping out of local optimum. Wang et al. designed an opposition based learning strategy where the candidates are transformed from one search space to another, which give the candidates more chance to find the global optimal solution [23]. Li et al. designed four different learning models by a self-learning approach and the particles are assigned with different role in the search process only based on their local fitness landscape [8]. Different from the traditional PSO, Cheng et al. employed the competition mechanism in which the particles are required to update their search patterns when they lose competition [2].

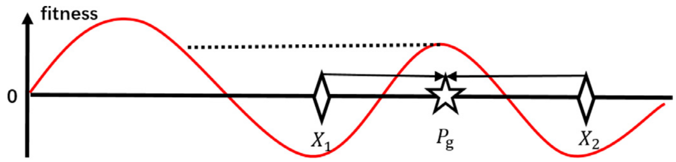
3. Adaptive learning based PSO algorithm (ALPSO)

As we know that all particles in original PSO learn from the global best particle to update their position and velocity until the termination condition is reached, even though the global best particle gets trapped into local optimum. This kind of learning mechanism makes the algorithm better in exploitation and have the feature of fast convergence but is invalid when dealing with the problems with complex search space. Now some representative variants of PSO are proposed, in which some strategies are used to restrict the learning object of particles to maintain the diversity of the population like LPSO, CLPSO. These strategies make the algorithm get good performance in exploration but will slow down the convergence rate of the algorithm.

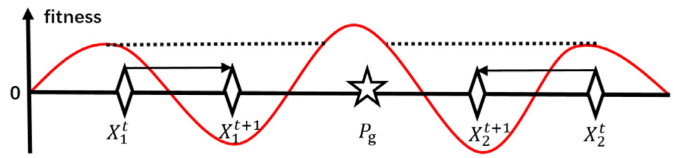
In order to balance between exploitation and exploration, the algorithm is proposed to obtain a better performance which is ALPSO. In ALPSO, we use a tolerance based search direction adjustment mechanism. The mechanism can make the swarm adjust their search direction to avoid falling into a local optimum adaptively and shrink the search space. Furthermore, we use a self-learning based candidate particle generation strategy to generate a candidate particle as a learning object of swarm, in which the swarm can do exploration in different areas within the search space. Then, for the purpose of ensuring the efficiency and accuracy of the algorithm, we use a potential prediction strategy to predict the potential ability of candidate particle to lead swarm do exploitation in different dimensions.

3.1. Tolerance based search direction adjustment mechanism (TSDM)

It is well known that the swarm in original PSO is more likely to get trapped into local optimum when searching in a complex space. For a certain dimension, the evolutionary state can be described by Fig. 1(a) where particles search for a maximum within one dimension and the red curve illustrates the variation trend of the fitness value in this dimension. P_g is the global best particle close to a local optimum. In Fig. 1(a), it is obviously that each particle, e.g., X_1 and X_2 may search along the direction to P_g and the swarm will fall into local optimum after several iterations. Suppose $f(P_i)^t$ denotes the fitness value of the ith particle's best position at the tth iteration. The ith particle's current position, i.e., X_i will be compared to



(a) P_g close to a local optimum



(b) P_g close to global optimum

Fig. 1. The evolutionary state of swarm in complex search space.

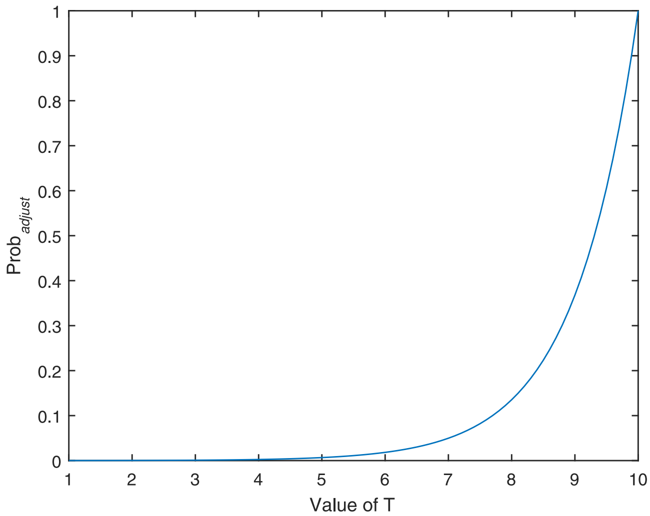


Fig. 2. Direction adjustment probability with T.

P_i after each iteration, P_i and $f(P_i)^t$ will be updated. The situation that all particles' best position are not improved in one iteration can be described as Eq. (2),

$$\sum_{i=1}^{n} f(P_i)^t - f(P_i)^{t-1} = 0 \tag{2}$$

where n is the size of the population, and t is the number of iteration. Obviously, the characteristic that all particles' best position don't get improved after one iteration may reflect that the swarm has got trapped into local optimum.

In order to avoid swarm falling into local optimum and guarantee the efficiency of algorithm, we can adjust the swarm's search direction when Eq. (2) is satisfied. However, we can not make a decision that adjust the search direction of swarm just rely on the situation described above which emerges once especially when swarm searches in a complex solution space. As Fig. 1(b) shows, particles search for a maximum within one dimension and P_g is the global best particle close to the global optimum. Particles in swarm at the tth iteration, e.g., X_1^t and X_2^t may search along the direction to P_g and these particles' best position will not be improved in the $t+1$th iteration, and in this case the Eq. (2) is satisfied too. However, in the long-term evolution of swarm, P_g has the potential ability to lead the swarm to improve and it is promising for swarm to search the global optimum in the next several iterations.

Therefore, for the purpose of avoiding that swarm gets trapped into local optimum and taking full advantage of P_g's evolutionary potential, we propose a tolerance based search direction adjustment mechanism (TSDM), which can help swarm comprehend the state of evolution during the iteration of algorithm to prevent the swarm from falling into local optimum and adjust the search direction at a proper time.

In fact, as the number of occurrences of the situation that all particles' best position are not improved increases, the probability that swarm gets trapped into local optimum will also increase, then the swarm need to adjust it's search direction timely. We denote a variable tolerance of the swarm by T, and T is used as a counter and initialized to 0. If all particles are not improved after one iteration, we can update the parameter T as the following Eq. (3).

$$T = T + 1 \tag{3}$$

It is obviously that, the swarm will be more likely to get into local optimum as the value of T becomes larger, which means the swarm should adjust it's search direction. We denote the probability that swarm adjust it's search direction as $Prob_{adjust}$, and $Prob_{adjust}$ is empirically obtained by the following tolerance Eq. (4).

$$Prob_{adjust} = \frac{\exp(T) - 1}{\exp(10) - 1} \tag{4}$$

Here $Prob_{adjust}$ is updated after each iteration. Fig. 2 shows how $Prob_{adjust}$ changes with T. If $Prob_{adjust}$ is larger than a random number belongs to [0, 1], the swarm will stop learning from the current P_g and adjust it's search direction to a new particle. Details of the procedure are shown as below in Algorithm 1.

Algorithm 1 Tolerance based search direction adjustment mechanism.

```
Initialize T = 0;
if (Σ_{i=1}^{n} f(P_i)^t − f(P_i)^{t−1} == 0) then
    T = T + 1;
end if
Generate a random number rand() between [0, 1];
if ((exp(T)−1)/(exp(10)−1) > rand()) then
    Stop the swarm from learning the current P_g
end if
```

As shown in the Algorithm 1 and Fig. 2, when the tolerance value of swarm, i.e., T is small, $Prob_{adjust}$ is determined by T and is more likely smaller than a random number which belongs to [0, 1], then swarm will still learn from current P_g. P_g's potential leading ability will be fully utilized in the next several iterations especially when P_g is close to the global optimum as shown in Fig. 1(b). If all particles are still not improved in the next several iterations, the value of T will increase continuously and the swarm is more likely to fall into a local optimum, then the value of $Prob_{adjust}$ is probably larger than a random number belongs to [0, 1] and the swarm will adjust it's search direction by learning a new particle. The new particle is called *Candidate* and the detail of generating a *Candidate* is presented in Section 3.2. Then the swarm may jump out of the current local optimum after learning from *Candidate* for a period of iterations. In summary, the strategy, i.e., TSDM can take full advantage of P_g's potential leading ability under the premise of helping swarm jump out of local optima.

3.2. Self-learning based candidate generation strategy

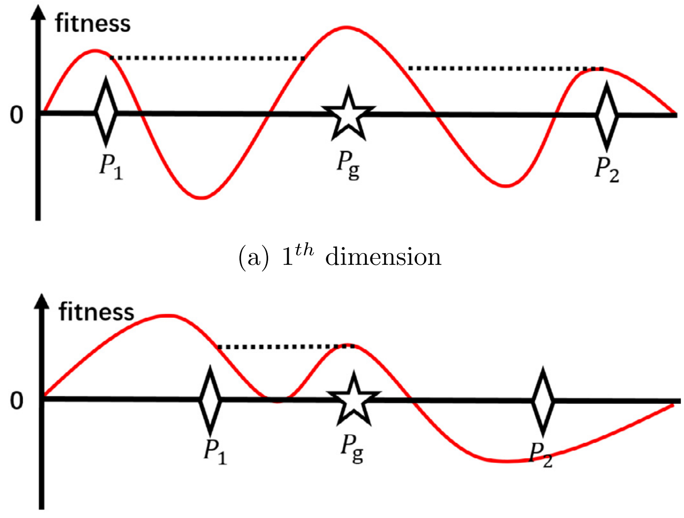
In ALPSO, all particle learn from P_g at the beginning, swarm use the TSDM to determine whether it need to adjust its search direction. When the swarm is likely trapped into a local optimum, it will adjust its search direction by learning from a new particle which presented as *Candidate*. It is easy to randomly generate a *Candidate* in the search space and swarm can learn from the generated *Candidate* to jump out of the current local optimum. However, the randomly generated *Candidate*'s ability to lead swarm to evolve may not be guaranteed especially when swarm searches in a complex space, it is more likely that the *Candidate* will lead swarm into another local optimum. In order to generate the *Candidate* effectively, we propose a self-learning based candidate generation strategy by which the swarm learns its own advantages and makes full use of every particle's excellent historical best structure of solution during each iteration of algorithm.

It is obviously that the current P_g's fitness value is still the best, and P_g still maintains good solution structure in most out of D dimensions. So P_g's solution structure is worth learning when generating the *Candidate*. In addition, because the fitness value of a particle is determined by the particle's solution structure of all D dimensions, which is represented as Eq. (5).

$$f(\boldsymbol{Particle}_i) = f(x_i^1, x_i^2, \ldots, x_i^D) \tag{5}$$

Therefore, the particle whose fitness value is slightly worse may have good solution structure in some particular dimensions, and it is also worth learning under this situation.

Fig. 3 shows a evolutionary state of swarm in two certain dimensions. Particles search for a maximum in each dimension and the red curve illustrates the variation trend of fitness value in each dimension. P_g is the current global best particle of swarm, P_1 and P_2 are two individual best position of X_1 and X_2, respectively. It can be seen that P_g has the best solution structure in 1th dimension but P_1 has a little better solution structure than P_g in 2nd dimension. Because the fitness value of a particle is determined by the particle's solution structure of all D dimensions, as Eq. (5) described, P_g will not be replaced by P_1 and we may neglect the good solution structure that P_1 holds in 2nd dimension, which is also worth learning when generating *Candidate*.



(b) 2^{nd} dimension

Fig. 3. The evolutionary state of swarm in two certain dimensions.

For the above reasons, *Candidate* will be generated not only according to the P_g but also the individual best solutions that all particles hold. And $Candidate(candidate^1, candidate^2, \ldots, candidate^D)$ is the *Candidate* particle with D dimensions. The detail of generating *Candidate* is described as follows in Algorithm 2.

Algorithm 2 Self-learning based candidate generation strategy.

```
for each dimension d from 1 to D do
   Generate a random number rand() between [0, 1];
   if Prob_Candidate > rand() then
      Candidate^d = P_g^d;
   else
      randomly select two P_k and P_m from the swarm;
      if f(P_i(k)) < f(P_i(m)) then
         Candidate^d = P_k^d + Gaussian(σ^d);
      else
         Candidate^d = P_m^d + Gaussian(σ^d);
      end if
   end if
end for
```

Here, $Prob_{Candidate}$ is the probability between [0, 1] that $Candidate^d$ will obtain the structure of P_g^d in dth dimension. Obviously, as the value of $Prob_{Candidate}$ increases, the generated *Candidate* will be more similar to P_g, on the contrary, the *Candidate* may be quite different from P_g and has the ability leading swarm jump out from current local optimum. Then σ^d is standard deviation variable which reflects the distribution of all particles' best solution, in the swarm with $\boldsymbol{n}$ particles it is obtained by the following Eqs. 6 and (7).

$$Average^d = \frac{1}{n}\sum_{i=1}^{n} P_i^d \tag{6}$$

$$\sigma^d = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i^d - Average^d)^2} \tag{7}$$

Considering that the goal of generating *Candidate* is to lead the swarm jump out from local optima, it may be inefficient to reconstructed the solution structure of *Candidate* randomly within the search space. To make sure that the reconstructed solution structure is not too bad, we randomly select two particles $P_i(k)$ and $P_i(m)$, and choose the more excellent one as an example with a Gaussian offset value determined by σ^d. As the process of selecting is random, theoretically all particles in the swarm can provide their own search information to the generating of *Candidate*. *Candidate* may get a superior solution structure to the current P_g and is more likely close to global optimum, so swarm may jump out of local optima by adjusting

Table 1
The 12 basic benchmark functions.

Functions	Domain	Name
$f_1(x)=\sum_{i=1}^{n} x_i^2$	[−100, 100]	Sphere
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[−10, 10]	Schwefels P2.22
$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{n}X_j\right)^2$	[−100, 100]	Quadric
$f_4(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	[−10, 10]	Rosenbrock's
$f_5(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	[−100, 100]	Step
$f_6(x)=\sum_{i=1}^{n}-x\sin(\sqrt{x_i})+418.9829n$	[−500, 500]	Schwefels
$f_7(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[−5.12, 5.12]	Rastrigin
$f_8(x)=\sum_{i=1}^{n}\left[y_i^2-10\cos(2\pi y_i)+10\right]$	[−5.12, 5.12]	Noncontinuous Rastrigin
$f_9(x)=20+e-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	[−32, 32]	Ackley
$f_{10}(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[−600, 600]	Griewank
$f_{11}(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_i)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_i-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	[−50, 50]	Penalized
$f_{12}(x)=0.1\left\{\sin^2(3\pi x_i)+\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	[−50, 150]	Penalized

it's search direction to *Candidate* and global optimum may be found after a period of iterations. In conclusion, this strategy ensures the algorithm's ability of exploration by utilizing all particles' individual best solution structure and reduces the effects on the rate of convergence at the same time.

3.3. Competitive learning based prediction strategy

As described above, the *Candidate* is randomly generated as a new learn object. So its ability to lead the swarm can't be guaranteed. It's reckless for the algorithm to replace the current P_g with *Candidate* without any proving on the ability of *Candidate*. In addition, the current P_g may still have a better structure of position than *Candidate* which is worth learning. To enhance the exploitation of algorithm and take advantage of the current P_g and *Candidate*, we propose a competitive learning strategy to predict the potential leading ability of *Candidate* in ALPSO.

After a *Candidate* has been generated, ALPSO will predict the potential leading ability by evaluating the performances of the swarm in which the particles learn from the current P_g and *Candidate* in the next time of iteration, respectively. The velocity of these two particles will update according to the following rules.

Once the swarm learns from the current P_g, the velocity will be updated by the following Eq. (8):

$$v_i^d=\omega v_i^d+c_1r_1^d(P_i^d-x_i^d)+c_2r_2^d(P_g^d-x_i^d) \tag{8}$$

And if the swarm learns from the *Candidate*, the velocity will be updated by the following Eq. (9):

$$v_i^d=\omega v_i^d+c_1r_1^d(P_i^d-x_i^d)+c_2r_2^d(Candidate^d-x_i^d) \tag{9}$$

There is a competitive relationship between the current P_g and *Candidate*, after one time iteration and the swarm will choose the better one as the new P_g in the next several iterations, and the worse particle will then learn from the better one to update the status of itself. This is why we name it as "competitive learning". In order to measure the ability leading swarm of P_g and *Candidate* conveniently, the Eq. (10) is described as follows,

$$\text{Competitiveness}_{\text{L}}=\sum_{i=i}^{n}f(x_i)^{t+1}-f(x_i)^t \tag{10}$$

where L stands for the learn object of the swarm. If $Competitiveness_{P_g}$ is larger $Competitiveness_{Candidate}$, it means the improvement when swarm learns from the current P_g is better, and vice versa. Then swarm will choose a better one as the new P_g in the next iteration. Details of this strategy is concluded as Algorithm 3.

3.4. Framework of ALPSO algorithm

Now, we can sum up the framework of this ALPSO algorithm in the following Algorithm 4.

Table 2
The 28 CEC'2013 benchmark functions.

	No.	Functions	Optimum
Unimodal functions	13	Sphere function	−1400
	14	Rotated high conditioned elliptic function	−1300
	15	Rotated bent cigar function	−1200
	16	Rotated discus function	−1100
	17	Different powers function	−1000
Multimodal functions	18	Rotated Rosenbrocks function	−900
	19	Rotated Schaffers F7 function	−800
	20	Rotated Ackleys function	−700
	21	Rotated Weierstrass function	−600
	22	Rotated Griewanks function	−500
	23	Rastrigins function	−400
	24	Rotated Rastrigins function	−300
	25	Non-continuous rotated Rastrigins function	−200
	26	Schwefel's function	−100
	27	Rotated Schwefel's function	100
	28	Rotated Katsuura function	200
	29	Lunacek-Bi-Rastrigin function	300
	30	Rotated Lunacek Bi-Rastrigin function	400
	31	Expanded Griewanks plus Rosenbrocks function	500
	32	Expanded Scaffers F6 function	600
Composition functions	33	Composition function 1 ($n = 5$,rotated)	700
	34	Composition function 2 ($n = 3$,unrotated)	800
	35	Composition function 3 ($n = 3$,rotated)	900
	36	Composition function 4 ($n = 3$,rotated)	1000
	37	Composition function 5 ($n = 3$,rotated)	1100
	38	Composition function 6 ($n = 5$,rotated)	1200
	39	Composition function 7 ($n = 5$,rotated)	1300
	40	Composition function 8 ($n = 5$,rotated)	1400

Algorithm 3 Competitive learning based prediction strategy.

1: **for** iteration from t to $t+1$ **do**
2:   **for** each particle i and i from 1 to n **do**
3:     $Competitiveness_{Candidate} += f(x^i)^{t+1} - f(x^i)^t$;
4:   **end for**
5: **end for**
6: **for** iteration from t to $t+1$ **do**
7:   **for** each particle i and i from 1 to n **do**
8:     $Competitiveness_{P_g} += f(x^i)^{t+1} - f(x^i)^t$;
9:   **end for**
10: **end for**
11: **if** $Competitiveness_{Candidate} > Competitiveness_{P_g}$ **then**
12:   Update P_g: $P_g = Candidate$;
13:   $T = 0$ // T is the tolerance counter of the swarm;
14: **else**
15:   P_g doesn't change;
16:   $T = T - 1$ // T is the tolerance counter of the swarm;
17: **end if**

3.5. Computational complexity of ALPSO

Compared with the basic PSO, the main difference of ALPSO is the *if* conditional statement (from line line 9 to line 14). Obviously, the Algorithm (1) is used to update the variable T and $Prob_{adjust}$, and the computational complexity of which is only (1). In the Algorithm 2, if the D dimensions solution structure of *Candidate* is generated, the computational complexity of generating a *Candidate* is $O(D)$. Suppose N is the size of the swarm, regarding to algorithm (3), which performs one iteration by applying swarm learn the P_g and *Candidate* respectively, its computational complexity is $O(N \cdot D)$. Therefore, the *if* conditional statement's computational complexity is $O(N \cdot D)$.

As a result, if the ALPSO's stop condition is a fixed iteration number presented as *Ite*, the entire computational complexity of ALPSO is $O(Ite \cdot N \cdot D)$.

Algorithm 4 Framework of ALPSO.

Initialize all particles' positions and velocities within the search space;
Initialize $T = 0$, $Prob_{adjust} = 0$;
Evaluate the fitness value of every particle;
Update P_i^t and P_g^t ;
while (stop condition in not reached) **do**
    Update all particles' X_i^t and V_i^t;
    Evaluate the fitness value of every particle;
    Update P_i^t and P_g^t;
    if $P > rand()$ **then**
        stop the swarm learning from the current P_g;
        Use the algorithm 2 to generate a **Candidate** particle;
        Use the algorithm 3 to choose a better particle as new P_g from the current P_g and the **Candidate**;
        Use the algorithm 1 to update T and $Prob_{adjust}$;
    end if
end while

Table 3
Parameters settings.

Algorithm	Parameters
GPSO	$\omega = 0.4, c_1 = c_2 = 2$
GPSO with decreasing weight	$\omega = [0.4, 0.9], c_1 = c_2 = 2$
LPSO	$\chi = 0.7298, c_1 = c_2 = 1.49445$
DMS-PSO	$\chi = 0.7298, c_1 = c_2 = 1.49445, M = 4, R = 10$
CLPSO	$\omega = [0.4, 0.9], c = 1.49445, gapM = 7$
ALPSO	$\omega = [0.4, 0.9], c_1 = c_2 = 2$

Table 4
Results comparison on 12 basic functions.

		ALPSO	GPSO $\omega = 0.4$	GPSO$\omega = [0.4, 0.9]$	LPSO	CLPSO	DMS-PSO
$f1$	MEAN	1.70E−93	**9.12E−162**	9.25E−51	4.57E−23	2.02E−29	7.68E−28
	SD	1.24E−91	**3.95E−161**	3.97E−50	1.02E−22	3.70E−29	1.62E−27
$f2$	MEAN	2.11E−28	**6.22E−91**	2.25E−29	2.82E−12	1.68E−17	3.28E−13
	SD	7.52E−27	**1.89E−90**	4.41E−29	9.54E−12	2.82E−17	1.24E−12
$f3$	MEAN	6.61E−08	**8.12E−11**	1.05E−01	1.38E+02	1.75E−03	1.38E+01
	SD	2.14E−07	**1.07E−10**	8.94E−02	9.08E+01	2.54E−03	1.19E+01
$f4$	MEAN	**1.78E+01**	1.98E+01	2.68E+01	4.20E+01	2.64E+01	2.32E+01
	SD	4.12E+00	**2.21E+00**	2.30E+01	2.71E+01	2.02E+01	9.60E−01
$f5$	MEAN	**0.00E+00**	1.83E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	SD	**0.00E+00**	7.86E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f6$	MEAN	**1.41E+03**	4.57E+03	2.65E+03	3.29E+03	2.59E+03	3.23E+03
	SD	**3.23E+02**	8.12E+02	6.39E+02	5.59E+02	4.67E+02	5.78E+02
$f7$	MEAN	**5.12E−14**	5.29E+01	3.20E+01	3.70E+01	3.92E−12	2.32E+01
	SD	**6.53E−14**	1.77E+01	1.31E+01	8.08E+00	3.88E−12	4.51E+00
$f8$	MEAN	**0.00E+00**	2.85E+00	4.50E−01	4.35E+00	**0.00E+00**	1.95E+00
	SD	**0.00E+00**	1.93E+00	7.40E−01	1.68E+00	**0.00E+00**	2.16E+00
$f9$	MEAN	1.04E−14	1.41E+00	**1.02E−14**	3.98E−11	1.11E−14	3.30E−14
	SD	1.07E−15	9.49E−01	**4.20E−15**	1.33E−10	3.15E−15	2.75E−14
$f10$	MEAN	**0.00E+00**	1.80E−02	2.04E−02	1.53E−02	1.62E−02	4.31E−03
	SD	**0.00E+00**	2.06E−02	1.84E−02	1.07E−02	1.37E−02	8.41E−03
$f11$	MEAN	**1.57E−32**	6.22E−02	2.07E−02	4.15E−02	3.99E−30	2.75E−28
	SD	**0.00E+00**	1.41E−01	4.15E−02	5.08E−02	1.29E−30	1.10E−27
$f12$	MEAN	**1.35E−32**	2.27E−01	4.39E−03	9.30E−03	1.94E−30	1.10E−03
	SD	**0.00E+00**	5.02E−01	5.38E−03	2.10E−02	3.98E−30	3.30E−03

4. Experimental study

4.1. Benchmark functions and parameter settings

In order to verify the performance of the ALPSO algorithm, here we use 40 test functions from two groups in our experiments. The first group has 12 basic functions ($f1$–$f12$), which are shown in Table 1. The first five functions ($f1$–$f5$) are unimodal functions and the next seven functions are multimodal functions. By doing experiments on these functions, we can verify that if the ALPSO can maintain the fast convergence feature and have the ability of dealing with multimodal

Table 5
Results comparison on 28 CEC'2013 functions.

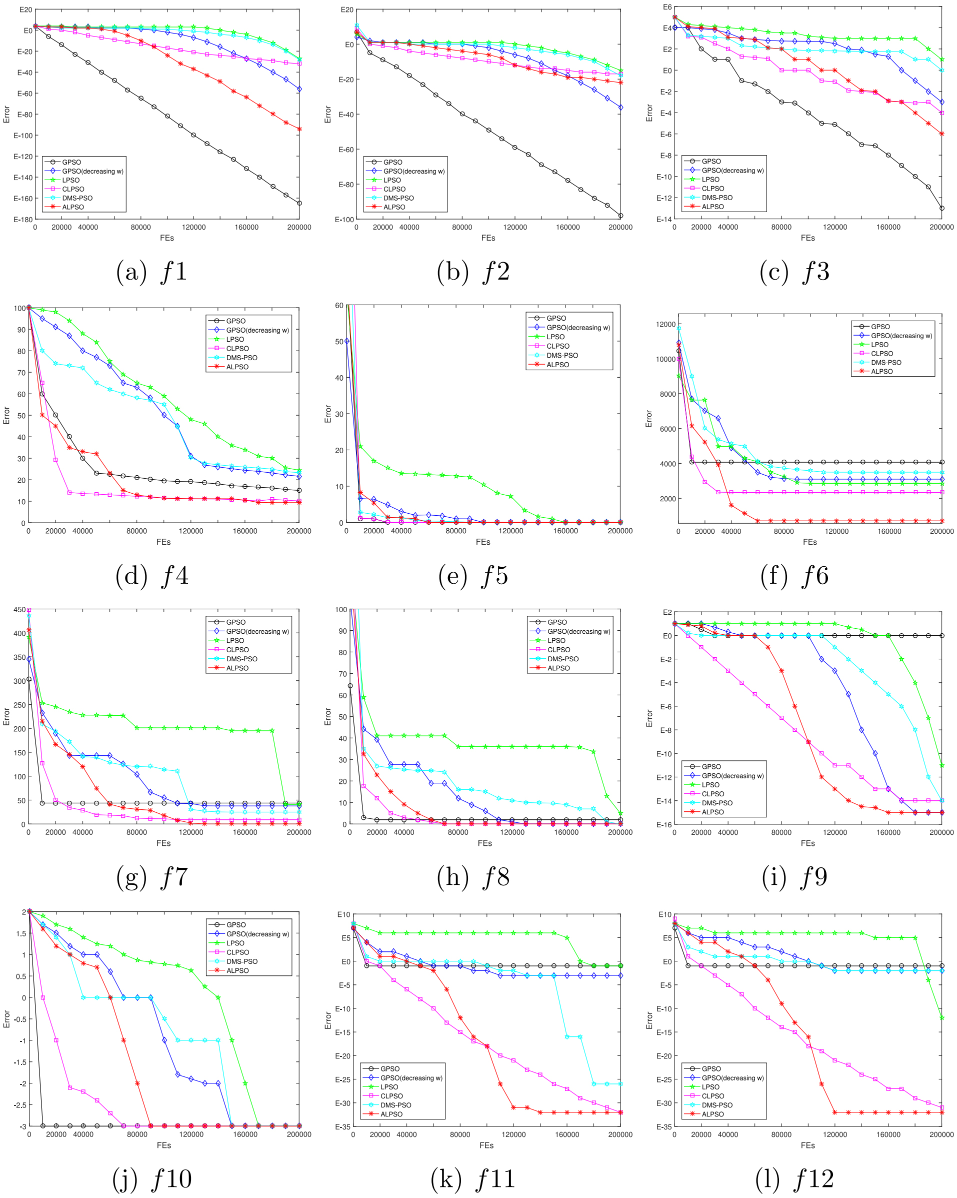
		ALPSO	GPSO $\omega = 0.4$	GPSO$\omega = [0.4, 0.9]$	LPSO	CLPSO	DMS-PSO
$f13$	MEAN	**2.09E−13**	9.68E+02	3.41E+03	1.18E+02	2.30E+02	2.84E−13
	SD	**1.13E−13**	1.42E+03	2.10E+03	2.38E+02	4.67E+02	1.29E−13
$f14$	MEAN	5.54E+06	**5.07E+06**	1.49E+07	1.88E+07	1.55E+07	6.77E+06
	SD	**2.19E+06**	6.36E+06	1.69E+07	2.33E+07	2.62E+07	2.23E+06
$f15$	MEAN	1.38E+09	2.00E+10	7.21E+10	2.18E+09	5.72E+09	**1.31E+08**
	SD	1.30E+09	1.56E+10	7.31E+10	2.09E+09	1.20E+10	**1.36E+08**
$f16$	MEAN	**9.85E+02**	2.50E+03	1.77E+04	6.98E+03	2.69E+03	1.04E+03
	SD	**2.57E+02**	1.52E+03	2.18E+04	1.18E+03	9.83E+02	3.11E+02
$f17$	MEAN	**2.27E−13**	9.38E+02	2.35E+03	9.24E+01	5.65E+01	1.34E+01
	SD	**6.23E−14**	9.30E+02	1.55E+03	1.02E+02	7.04E+01	3.82E+01
$f18$	MEAN	**4.16E+01**	1.14E+02	3.26E+02	6.76E+01	5.72E+01	5.22E+01
	SD	**1.68E+01**	8.28E+01	3.30E+02	1.99E+01	2.39E+01	1.54E+01
$f19$	MEAN	1.37E+02	1.49E+02	1.57E+02	6.65E+01	1.03E+02	**3.49E+01**
	SD	5.40E+01	4.40E+01	6.21E+01	4.70E+01	4.51E+01	**2.01E+01**
$f20$	MEAN	**20.893**	20.934	20.944	20.937	20.915	21.052
	SD	0.050	0.055	0.048	**0.035**	0.058	0.070
$f21$	MEAN	20.059	27.906	21.639	20.927	24.881	**16.139**
	SD	**2.268**	3.761	3.052	3.723	4.670	3.311
$f22$	MEAN	**2.16E+00**	2.64E+02	9.56E+02	1.14E+02	3.40E+01	1.18E+01
	SD	**4.48E+00**	3.17E+02	6.83E+02	7.22E+01	4.50E+01	3.23E+01
$f23$	MEAN	**3.53E−11**	1.18E+02	1.57E+02	5.67E+01	2.10E+01	2.92E+01
	SD	**1.49E−10**	2.97E+01	5.61E+01	1.33E+01	1.39E+01	7.33E+00
$f24$	MEAN	1.29E+02	1.43E+02	1.42E+02	1.56E+02	1.01E+02	**5.92E+01**
	SD	3.37E+01	4.65E+01	4.51E+01	5.99E+01	2.89E+01	**2.51E+01**
$f25$	MEAN	2.02E+02	2.30E+02	2.18E+02	2.03E+02	1.80E+02	**9.19E+01**
	SD	3.62E+01	4.99E+01	4.40E+01	**2.16E+01**	2.97E+01	2.59E+01
$f26$	MEAN	**3.15E+02**	2.61E+03	2.48E+03	2.53E+03	1.76E+03	1.56E+03
	SD	**8.59E+01**	5.59E+02	5.64E+02	9.12E+02	5.18E+02	3.49E+02
$f27$	MEAN	**3.20E+03**	4.23E+03	5.67E+03	6.72E+03	4.31E+03	3.41E+03
	SD	5.41E+02	6.13E+02	1.52E+03	**3.47E+02**	6.71E+02	6.93E+02
$f28$	MEAN	1.491	2.100	2.137	2.210	**1.315**	1.901
	SD	0.373	0.460	0.387	0.390	**0.358**	0.779
$f29$	MEAN	**33.71**	104.84	71.89	90.08	52.44	83.22
	SD	**3.919**	27.35	23.95	12.32	7.94	20.25
$f30$	MEAN	**30.00**	103.81	75.60	83.65	39.24	63.32
	SD	**1.61E−08**	23.42	52.52	16.11	4.06	9.96
$f31$	MEAN	**2.77E+00**	6.28E+02	4.48E+02	5.71E+00	4.27E+00	6.04E+00
	SD	**8.79E−01**	1.41E+03	1.42E+03	1.93E+00	5.21E+00	1.58E+00
$f32$	MEAN	11.012	11.189	11.420	11.338	11.606	**9.165**
	SD	**0.477**	0.587	0.536	0.601	0.683	0.909
$f33$	MEAN	**3.27E+02**	3.71E+02	7.09E+02	4.47E+02	3.31E+02	3.35E+02
	SD	**6.5E+01**	1.29E+02	2.48E+02	1.06E+02	9.9E+01	8.5E+01
$f34$	MEAN	**5.82E+02**	3.10E+03	2.57E+03	2.68E+03	1.88E+03	1.34E+03
	SD	**2.31E+02**	7.87E+02	9.01E+02	1.14E+03	6.09E+02	4.07E+02
$f35$	MEAN	4.20E+03	5.52E+03	5.07E+03	6.80E+03	5.38E+03	**3.60E+03**
	SD	7.25E+02	1.01E+03	1.21E+03	**5.43E+02**	5.98E+02	5.58E+02
$f36$	MEAN	2.77E+02	2.83E+02	2.80E+02	2.70E+02	2.27E+02	**2.48E+02**
	SD	1.62E+01	2.30E+01	1.28E+01	**7.65E+00**	1.01E+01	9.97E+00
$f37$	MEAN	2.95E+02	3.08E+02	3.02E+02	2.94E+02	2.86E+02	**2.76E+02**
	SD	**5.99E+00**	1.29E+01	1.36E+01	9.63E+00	1.05E+01	1.05E+01
$f38$	MEAN	**2.32E+02**	3.53E+02	3.52E+02	2.58E+02	2.33E+02	2.69E+02
	SD	**6.31E+01**	7.99E+01	7.93E+01	7.66E+01	6.52E+01	6.90E+01
$f39$	MEAN	9.17E+02	1.09E+03	9.81E+02	1.13E+03	9.47E+02	**6.95E+02**
	SD	6.07E+01	**5.85E+01**	1.21E+02	1.41E+02	1.08E+02	1.16E+02
$f40$	MEAN	**3.00E+02**	1.59E+03	2.21E+03	1.13E+03	5.42E+02	4.11E+02
	SD	**8.66E−13**	7.29E+02	4.41E+02	4.06E+02	3.79E+02	3.27E+02

functions. The second group has 28 functions ($f13$–$f40$) shown in Table 2 which are taken from the CEC'2013 test suit. Details of these functions can be find in the report [12]. The first five functions ($f13$–$f17$) are unimodal functions, the next 15 functions ($f18$–$f32$) are multimodal functions and the last 8 functions ($f33$–$f40$) are composition functions. Every functions' search space is complex, and we can testify the comprehensive performance of ALPSO by the experimental study on these functions.

The global optima of these functions shown in Table 1 are 0, the *Domain* is search range of every dimension.

In $f8$, the y_i is update by $y_i = \begin{cases} x_i, |x_i| \leq 0.5 \\ \frac{round(2x_i)}{x_i}, |x_i| \geq 0.5 \end{cases}$.

In $f11$, the y_i is update by $y_i = 1 + \frac{1}{4}(x_i + 1)$.



(a) $f1$ (b) $f2$ (c) $f3$

(d) $f4$ (e) $f5$ (f) $f6$

(g) $f7$ (h) $f8$ (i) $f9$

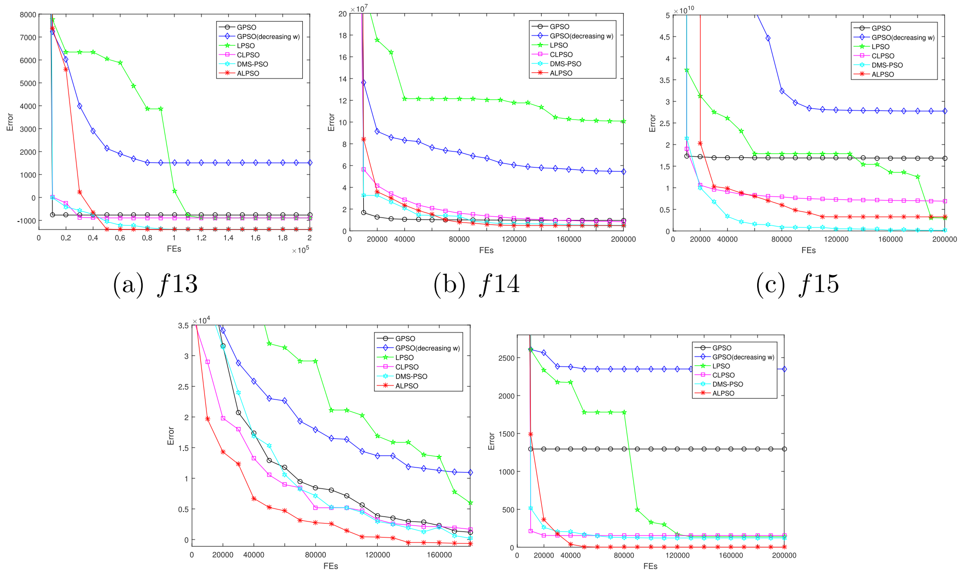
(j) $f10$ (k) $f11$ (l) $f12$

Fig. 4. Convergence process on 12 basic benchmark functions.

In $f11$ and $f12$, $u(x_j, a, k, m)\begin{cases} k(x_j - a)m, x_j > a \\ 0, -a \le x_j \le a \\ k(-x_j - 1)m, x_j < -a \end{cases}$.

To validate the performances of the ALPSO, five representative variants of PSO are chosen to compare with ALPSO. The first one is the global version PSO (GPSO), in the GPSO particles always learn from the P_g, so it has the feature of fast convergence when dealing with unimodal problem. The second one is the global version PSO which is designed to improve the swarm's ability of global search, and the inertia weight is linearly decreased from 0.9 to 0.4. The third one is local version of PSO (LPSO) using a ring topology. The fourth one is DMS-PSO, and the fifth one is CLPSO. The parameter settings of each algorithm are shown below in Table 3. The parameter M and R in DMS-PSO stands for the size of subswarm and regrouping period, respectively. The parameter $gamM$ in CLPSO is the refreshing gap of a particle.

To make it fair when comparing experimental results of using different algorithms, each algorithm will run on the corresponding benchmark function independently 50 times and the mean error of results will be displayed in the tables of results comparison shown below. Here, the dimension of all benchmark functions D is set as 30. The population size N in each algorithm is set as 20. The maximum of the fitness evaluation FEs number is 2×10^5.



(a) $f13$ (b) $f14$ (c) $f15$

(d) $f16$ (e) $f17$

Fig. 5. Convergence process on 5 unimodal functions of CEC'2013.

4.2. Results comparison on convergence accuracy

The comparison results on convergence accuracy including mean error (*MEAN*) and error's standard deviation (*SD*) which are listed in Tables 4 and 5. Here, we mark the best results performed by those algorithms on each test functions with bold font, and mark the second best results of them with underlined.

As the value of *MEAN* shown in Tables 4 and 5, we can obviously see that GPSO performs well on some basic unimodal functions ($f1$–$f3$), but gets bad results when dealing with multimodal functions and the unimodal functions with complex search space on CEC'2013 test suit. A possible reason is that, since the inertia weight is linearly decreased in GPSO, it helps GPSO adjust the balance between the exploration and exploitation and have a good convergence when dealing with unimodal problems; while for multimodal problems, the linearly changed weight can't adjust well along the evolving process and get good results. And for LPSO, it has the ability of maintaining the diversity of population by producing restrictions on global search, the structure of particles may not suit all test functions, so it can only performs well on some specific functions (i.e., $f5$, $f9$, $f19$).

As we know that CLPSO and DMS-PSO have a competitive capability on solving multimodal problems and they perform well on several test functions shown in Tables 4 and 5. Especially for DMS-PSO, it can get a second best results on several multimodal test functions, i.e.$f4$, $f10$, $f18$, $f22$, $f26$, $f27$, $f34$ and $f40$, it even gets results that superior to ALPSO on some multimodal test functions, i.e., $f19$, $f24$, $f25$ and $f32$. A possible reason may be that the competitive learning based strategy used in ALPSO wastes some fitness evolutions since the search direction of swarm needs to be constantly adjusted when swarm frequently gets trapped into local optima. On the contrary, as the results shown in Tables 4 and 5, DMS-PSO performs poorer on 5 basic unimodal functions and 5 unimodal test functions than ALPSO. The reason is that, DMS-PSO use a multi-swarm strategy to maintain the diversity of swarm within the entire search space, which has restriction on the convergence speed of DMS-PSO.

As mentioned above, ALPSO can jump out from local optima and get the ability of global search by TSDM and the self-learning based candidate generation strategy, which result in that swarm may adaptively adjust it's search direction to a more excellent particle and stop falling into local optima especially when swarm searches in a complex space. Moreover, ALPSO can get the ability of local search by the mechanism of competitive learning based prediction strategy, which chooses P_g with more leading ability. As the Tables 4 and 5 shows, ALPSO gets the most favorable results on 6 out of 7 basic multimodal test functions and gets the best or the second best results on 12 out of 15 CEC'2013 multimodal test functions, except for $f19$, $f24$ and $f25$, which means ALPSO has an excellent performances when dealing with multimodal test functions. Therefore, the efficiency of ALPSO to deal with multimodal test functions is verified.

Simultaneously, from Tables 4 and 5, we can see that ALPSO is not restricted when dealing with simple unimodal test functions. The reason is that there is no need for swarm in ALPSO to frequently adjust the search direction as the swarm's evolutionary state which is determined by TSDM is fine. In other words, the strategies utilized in ALPSO produce little impact on the algorithm's convergence speed when algorithm solves simple unimodal test functions, which resulting in that ALPSO also performs well on all unimodal test functions described above (i.e., $f1$–$f5$, $f13$–$f17$). As for the results on

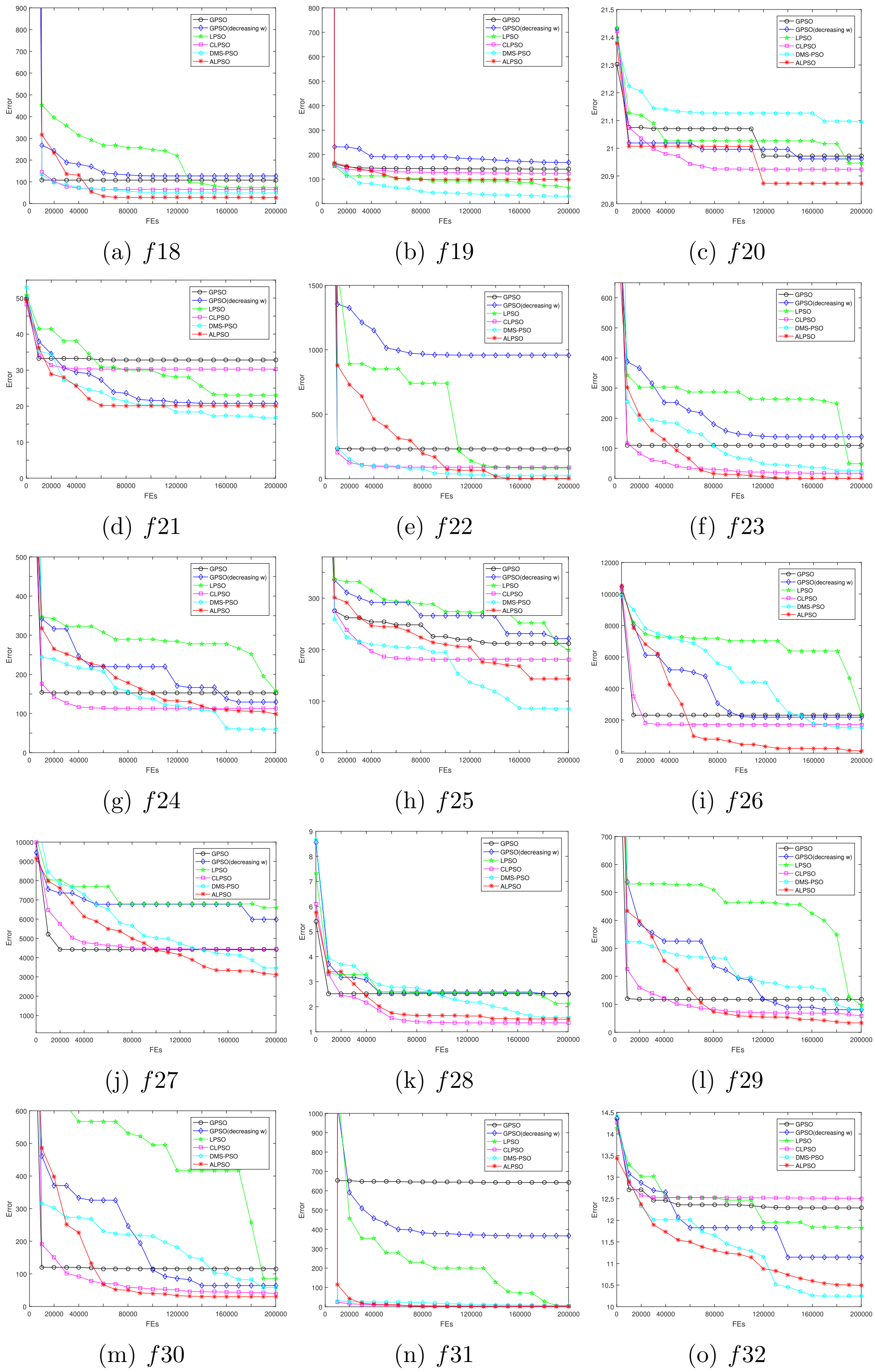


Fig. 6. Convergence process on 15 basic multimodal functions of CEC'2013.

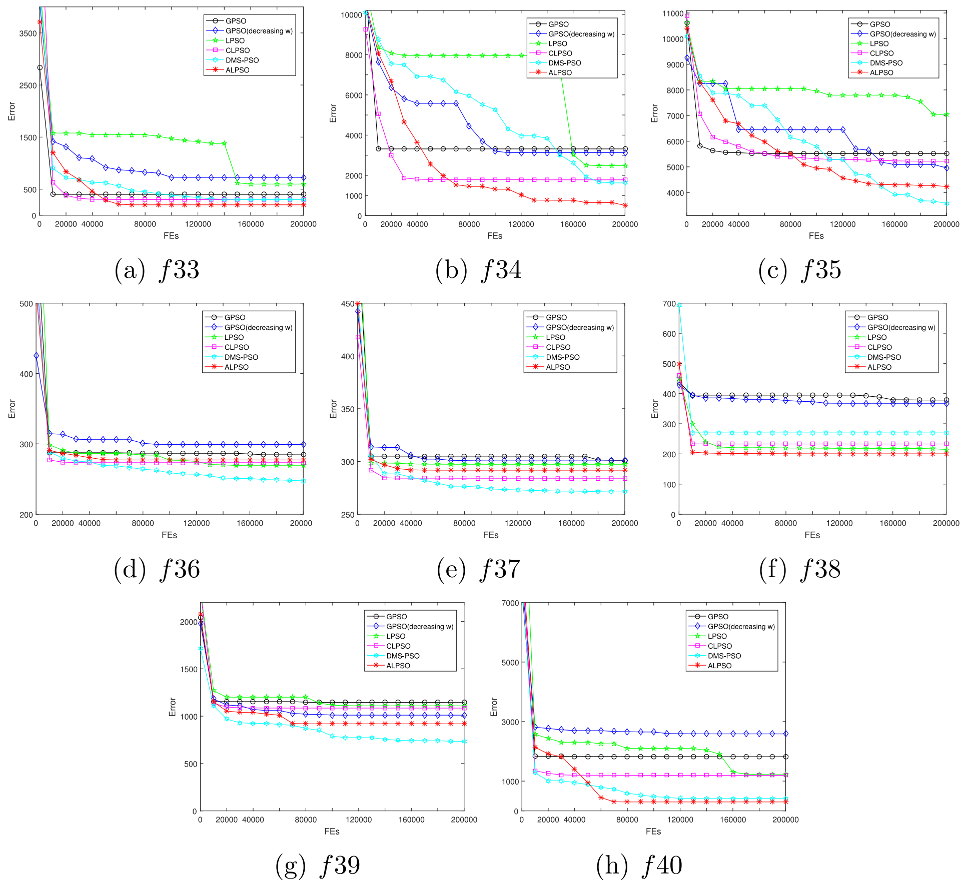


Fig. 7. Convergence process on 8 composition functions of CEC'2013.

composition functions ($f33$–$f40$) which are shown in Table 5, ALPSO gets the best results or the second best results on 6 out of 8 composition functions, which presents the performance of ALPSO to solve more tough problems.

In addition, as the value of *SD* shown in Tables 4 and 5, we can see that the *SD*'s values of ALPSO on most functions ($f5$–$f12$, $f13$–$f18$, $f21$–$f23$, $f26$–$f34$, $f37$–$f40$) is much smaller than others, which means that the performance of our proposed ALPSO is steady in convergence process.

4.3. Results comparison on convergence speed

We then conduct experiments on the convergence speed of the same 40 benchmark test functions to further testify the convergence speed of ALPSO. Figs. 4–7 show the convergence process of the algorithms.

From the above figures, we can see that the proposed ALPSO can converge with an ideal convergence speed, especially on multimodal functions. And from the convergence process figures of the 12 basic test functions, it can also be find that ALPSO can converge to the global optimal value on the multimodal functions in the first group(from $f5$ to $f12$), and has the fastest convergence speed among these algorithms. This might because that, in ALPSO, once the swarm get trapped into a local optimum, the swarm will timely adjust its search direction by TSDM, and this mechanism help ALPSO jump out from local optima and accelerate the searching speed especially when ALPSO dealing with these multimodel functions within a complex search space. And from Fig. 5 –7, we can also find that, ALPSO still have a fast convergence rate and the speed of convergence ranked first on many other functions in the second group (i.e. $f13$, $f14$, $f16$, $f17$, $f18$, $f20$, $f21$, $f23$, $f26$, $f27$, $f30$–$f40$).

From the above results, we can now conclude that ALPSO can get better results than the other five algorithms, not only on the solution accuracy but also the convergence speed, especially when dealing complex problems.

5. Conclusion and future work

In this paper, we present a hybrid PSO algorithm based on adaptive learning strategy, namely ALPSO to alleviate the premature convergence of PSO on many complex problems. We mainly focus on the improvement of the swarm structure learning and particles' local search strategy learning. As we know, the main reason that the basic PSO usually gets trapped in local optima is the particles in the population mostly learn from the global best. Here, we employ a self-learning based candidate generation strategy which generates a new candidate not only utilize the information from global best particle but also the historical individual best particles in all dimensions. This strategy contributes to the improvement on the exploration ability of our new algorithm. At the same time, we also propose a competitive learning based prediction strategy to help choose the best candidate in the evolution process, which can help to find those particles with potential ability to lead the swarm to find the global best and retain them in the population. As a result of this, the exploitation ability can also be improved. Furthermore, for the sake of getting a good balance between exploration and exploitation, we design a tolerance based search direction adjustment mechanism, which help decide the proper timing of the particles change the search direction in the solution space. We conduct the experimental study on 40 test functions, including 12 basic test functions, 20 test functions and 8 composition functions from CEC'2013 Test Suite. The results of this ALPSO algorithm comparing with 5 state-of-the-art algorithms show it can get better convergence accuracy as well as faster convergence speed in most cases.

The experimental results show the competitive performance of ALPSO and the ability to solve complex problems such as multimodal test problems and composition test problems is verified. As the problems in reality are getting more and more complicated such as financial optimization problems and scheduling optimization problems, in the future, we will further go on the theoretical analysis of the parameter settings in ALPSO, and try to investigate the applications of this ALPSO algorithm into solving various challenging optimization problems in reality.

Acknowledgment

This work is supported by National Natural Science Foundation of China [Grant nos. 61773296, 61573157]; Research Fund for Academic Team of Young Scholars at Wuhan University (Project no. Whu2016013).

References

[1] A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Comput. Oper. Res. 33 (3) (2006) 859–871.
[2] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2015) 191–204.
[3] A. Glvez, A. Iglesias, A new iterative mutually coupled hybrid GA-PSO approach for curve fitting in manufacturing, Appl. Soft Comput. J. 13 (3) (2013) 1491–1504.
[4] D.W. Gong, J. Sun, Z. Miao, A set-based genetic algorithm for interval many-objective optimization problems, IEEE Trans. Evol. Comput. PP (99) (2016). 1–1
[5] N. Kalaiarasi, S. Paramasivam, S.S. Dash, P. Sanjeevikumar, L. Mihet-Popa, PSO based MPPT implementation in dspace controller integrated through z-source inverter for photovoltaic applications, Energies 9 (2017) 1–10.
[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, 2002, pp. 1942–1948.
[7] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the Congress on Evolutionary Computation, 2002, pp. 1671–1676.
[8] C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, IEEE Trans. Syst. Man Cybern. Part B Cybern. 42 (3) (2012) 627–646.
[9] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Comput. Secur. 72 (2018) 1–12.
[10] P. Li, J. Li, Z. Huang, T. Li, C.-z. Gao, W.-B. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, Cluster Comput. (2017) 1–10.
[11] P. Li, J. Li, Z. Huang, T. Li, C.-z. Gao, S.-M. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Gener. Comput. Syst. 74 (2017) 76–85.
[12] J. Liang, B. Qu, P. Suganthan, A. Hernndez-Daz, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013 Technical Report 201212.
[13] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[14] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 2006, pp. 9–16.
[15] Y. Liu, D. Gong, S. Jing, Y. Jin, A many-objective evolutionary algorithm using a one-by-one selection strategy, IEEE Trans Cybern. 47 (9) (2017) 2689–2702.
[16] T.K. Maji, P. Acharjee, Multiple solutions of optimal PMU placement using exponential binary PSO algorithm for smart grid applications, IEEE Trans. Ind. Appl. 53 (3) (2017) 2550–2559.
[17] V.K. Pathak, A.K. Singh, A particle swarm optimization approach for minimizing GDT error in additive manufactured parts: PSO based GDT minimization, Int. J. Manuf. 7 (3) (2017) 69–80.
[18] A. Ratnaweera, S. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, Evolut. Comput. IEEE Trans. 8 (3) (2004) 240–255.
[19] A. Sedki, D. Ouazar, Hybrid particle swarm optimization and differential evolution for optimal design of water distribution systems, Adv. Eng. Inf. 26 (3) (2012) 582–591.
[20] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1999, pp. 1945–1950.
[21] Y. Tang, Z. Wang, J.A. Fang, Feedback Learning Particle Swarm Optimization, Elsevier Science Publishers B. V., 2011.
[22] F. Wang, Y. Zhang, Q. Rao, K. Li, H. Zhang, Exploring mutual information-based sentimental analysis with kernel-based extreme learning machine for stock prediction, Soft Comput. 21 (12) (2017) 3193–3205.
[23] H. Wang, S. Hui, C. Li, S. Rahnamayan, J.-s. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2) (2013) 119–135.
[24] H. Wang, W. Wang, H. Sun, S. Rahnamayan, Firefly algorithm with random attraction, Int. J. Bio Inspired Comput. 8 (1) (2016) 33–41.
[25] H. Wang, W. Wang, X. Zhou, H. Sun, J. Zhao, X. Yu, Z. Cui, Firefly algorithm with neighborhood attraction, Inf. Sci. 382C383 (2017) 374–387.
[26] S. Wang, Y. Zhang, Z. Dong, S. Du, G. Ji, C. Feng, C. Feng, C. Feng, C. Feng, P. Phillips, Feed-forward neural network optimized by hybridization of PSO and abc for abnormal brain detection, Int. J. Imaging Syst. Technol. 25 (2) (2015) 153–164.

[27] H. Wu, L. Kuang, F. Wang, R. Qi, G. Maoguo, L. Yuanxiang, A multiobjective box-covering algorithm for fractal modularity on complex networks, Appl. Soft. Comput. 61 (2017) 294–313.
[28] Z.H. Zhan, J. Zhang, Y. Li, S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1362–1381.
[29] X. Zhou, H. Wang, M. Wang, J. Wan, Enhancing the modified artificial bee colony algorithm with neighborhood search, Soft Comput. 21 (10) (2015) 1–11.
[30] X. Zhou, Z. Wu, H. Wang, S. Rahnamayan, Gaussian bare-bones artificial bee colony algorithm, Soft Comput. Fusion Found. Methodol. Appl. 20 (3) (2016) 907–924.