Particle Swarm Optimization Algorithm with Reverse-Learning and Local-Learning Behavior

Xuewen Xia
National Engineering Research Center for Satellite Positioning System, Wuhan University, China
Hubei Engineering University, Xiaogan, China
Email: laughkid@163.com, xwxia@whu.edu.cn

Jingnan Liu
National Engineering Research Center for Satellite Positioning System, Wuhan University, China

Yuanxiang Li
School of Computer, Wuhan University, Hubei Wuhan 430079, China

***Abstract*—In order to resolve conflict between convergence speed and population diversity of particle swarm optimization (PSO) algorithm, an improved PSO, called reverse-learning and local-learning PSO (RLPSO) algorithm, is presented in which a reverse-learning behavior implemented by some particles while local-learning behavior adopted by elite particles in each generation. During the reverse-learning process, some inferior particles of initial population and each particle's historical worst position are reserved to attract a particle to leap out of local optimums. Furthermore, the Hamming distance between the inferior particles is set to no less than a default rejection distance the aim of which is to maintain the diversity of population and improve RLPSO's exploration ability. In each generation, the difference between the best particle and the second-best particle is used to guide the best one to carry out local search which is crucial for improving RLPSO's exploitation ability. The results of experiments show that RLPSO has a good global searching ability and convergence speed especially in high dimension function.**

***Index Terms*—particle swarm optimization, reverse learning, local learning, premature convergence**

I. Introduction

Particle swarm optimization (PSO) algorithm is a global optimization method that originally developed by Kennedy and Eberhart in 1995[1]. Unlike other evolutionary computation algorithms, in which some genetic operations such as selection, crossover and mutation operators are adopted to manipulate a new individual, PSO searches for an optimum by each particle's flying through the problem space. This search mechanism enables PSO has high operating efficiency, fast convergence speed, and implementation simplicity.

Owing to its outstanding characters, PSO has been an effective tool for solving global optimization problems during the past two decades, such as non-linear function optimization, signal processing[2], image reconstruction [3], engineering optimization [4], etc. Although PSO has obtained some promising results in many research fields, many experiments also indicate that the traditional PSO algorithm easily falls into local optima while solving complex multimodal problems. Aiming at the issue, many researchers have made great effort during the past decade which will be discussed in Section II.

Based on previous works, a novel algorithm, called reverse-learning and local-learning particle swarm optimizer (RLPSO), is proposed in this paper. In RLPSO, some new features are proposed. First, much useful information within in some weak particles as well as in the best particle is taken into account to prevent population from prematurity. Second, a simple differential operator on some elite particles is adopted to enhance the exploitation ability of PSO.

The rest of this paper is organized as follows. Section II describes the framework of traditional PSO and reviews some improved PSOs. The detail of RLPSO algorithm is presented in Section III. Section IV describes the experimental study on RLPSO and comparison of 3 various improved PSO algorithm. Finally, conclusions are given in Section V.

II. Related Works

A. PSO

Similar to other swarm intelligence heuristic algorithm, PSO is a population-based stochastic optimization technique. In PSO, a continuous process of optimization is described as each particle's flying while is described as population's evolution in genetic algorithm (GA).

When searching in a D-dimensional hyperspace, the i^{th} particle in PSO has a velocity vector $\boldsymbol{V_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ and a position vector $\boldsymbol{X_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ to indicate its current state, where i is a positive integer indexing the particle in the swarm and D is the dimensions of the problem under study. The position vector $\boldsymbol{X_i}$ is regarded as a candidate solution of the problem while the velocity vector $\boldsymbol{V_i}$ is treated as a particle's search direction and step. During the process of optimization (or flying), each

Corresponding author: Xuewen Xia, email: laughkid@163.com, xwxia@whu.edu.cn

doi:10.4304/jsw.9.2.350-257

particle decides its trajectory according to its personal historical best position vector $pbest_i = [pb_{i1}, pb_{i2},\ldots, pb_{iD}]$ (i.e., *pbest* location) and the global best-so-far position vector $gbest = [gb_{i1}, gb_{i2},\ldots,gb_{iD}]$ (i.e., *gbest* location). The update rules of a particle's velocity and position are very simple, which are defined as

$$v_{ij}^{t+1} = \omega \cdot v_{ij}^{t} + c_1 \cdot r_1 \cdot \left(pb_{ij}^{t} - x_{ij}^{t} \right) + c_2 \cdot r_2 \cdot \left(gb_{j}^{t} - x_{ij}^{t} \right) \quad (1)$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t} \quad (2)$$

where ω is the inertia weight; $c1$ and $c2$ are the acceleration coefficients that determine the relative learning weight for $pbest_i$ and *gbest*, which called self-cognitive and social influence, respectively; $r1$ and $r2$ are two random numbers that are uniformly distributed over [0, 1]; i represents the current particle; and j represents the current dimension.

B. Some Modified PSO

Although PSO has been successfully applied in various fields since it was first introduced, especially in some complex, large scale, nonlinear and non-differentiable optimization problems, there are still many deficiencies in PSO algorithm, such as low accuracy of resolution, premature convergence and curse of dimensionality, etc. In order to solve above-mentioned problems, several modifications have been introduced to improve the performance of the traditional PSO during the past decade.

From the aforementioned parameters in PSO we know that the inertia weight ω and the acceleration constants c play important roles in optimization process. It is a simple and effective strategy to improve the efficiency of PSO that selecting a proper set of parameters or introducing a useful parametric adaptation mechanism. For example, in order to regulate swarm's convergence speed and global searching ability, Shi and Eberhart adopted a method of linearly deceasing ω with the iteration for PSO in [5] and then introduced a fuzzy adaptive ω method for PSO in [6]. In [7], the adjustment of ω not only depends on the current iterations of evolution but also relies on the current population diversity. Ratnaweera *et al.* [8] developed a self-organizing H-PSO with time-varying acceleration coefficients. After making stability and astringency analysis for traditional PSO, a set of optimized parameters was proposed by Trelea and M. Jiang in [9][10], respectively.

The population topology has a significant effect on the performance of PSO. There are several common topologies, such as ring, wheels, stars and Von Neumann neighbor structure, have been widely used in many variants of PSO [11]. After that, some topology tuning strategies have been proposed basing on the common topologies. In [12], a fully informed PSO, called FIPS, was proposed by Mendes *et al.,* in which a particle uses a stochastic average of *pbest*'s from all of its neighbors instead of using its own *pbest* and the *gbest* in the update equation. In [13], Euclidian distance between particles is deemed as a criterion to select proper *pbest* and *gbest* for a specific particle. In addition, many dynamic topologies, in which the static topologies have been replaced by some dynamically adjusted neighbor models, were introduced in [14][15] to avoid deficiencies of fixed neighborhoods. These modifications on topology improve the local searching ability and overcome the premature phenomena of PSO in deferent degree.

The choice of which learning pattern to use is very important for PSO to achieve a good results. Liang *et al.* [16] developed a comprehensive learning PSO (CLPSO) for multimodal problems. In CLPSO, a particle uses different particles' historical best information to update its velocity, and for each dimension, a particle can potentially learn from a different exemplar. Changhe Li *et al*.[17] presented a self-learning particle swarm optimizer (SLPSO), in which each particles has four candidates learning-patterns to cope with different situations in the search space. In [18], a common single *gbest* is replaced by a group of elite particles. And a rejection strategy is adopted to enable the elites widely distributed in the search space. In [19], a novel Baldwin effects based learning strategy is adopted to improve the performance of PSO, in which the historical beneficial information is utilizes to increase the potential search range and retains diversity of the particle population.

Although the modifications of PSO can generally be categorized into the aforementioned types, it should be pointed out that some complicated PSO variants may adopt multiple strategies above-mentioned simultaneously. The improvement strategy adopted in this paper involves parameter adjustment and learning pattern adjustment.

III. RLPSO ALGORITHM

As introduced above, the main aims of these strategies that adopted by PSO are to maintain population diversity and to improve global search ability with a high convergence speed. In this paper, a reverse learning strategy is selected to maintain population diversity while a local research strategy is adopted by *gbest* to improve the convergence speed. The details of the strategies are described as follow, respectively.

A. Reverse Learning of Moderate Particles

So far, most PSO algorithms use a similar learning pattern for all particles, in which each particle in a swarm learns from its own best historical experience and its neighborhood's best historical experience. This monotonic learning pattern may cause the swarm lack of intelligence to deal with different complex situations. The monotonic learning pattern is widely applied in most PSO because a hypothesis that there is little useful information in an inferior particle. However, it's just as true that an inferior particle may have good values on some dimensions of the solution vector though the particle has the worst values on other dimensions. The reason why the performance of elite particles is better than the performance of inferior particles is that the elite particles have good values on more dimensions than the

inferior particles, or the solution is vulnerable to the dimensions on which the elite particles acquire good values. So it is an arbitrary decision to regard that there is little useful information embedded in an inferior particle. In addition, there is usually a long distance between an inferior particle and a suboptimal particle. This trait of the inferior particles may be employed to drag some particles to jump out of local optimal. Hence, how to discover more useful information embedded in the inferior particles and thus how to utilize the information to improve PSO's global search ability are important for us.

In this paper, when the optimum information of the swarm is stagnant during the search process in a *D*-dimensional hyperspace, *n* particles will participate in a reverse learning. An inferior position in the initial population $W_k^0 = \left(w_{k1}^0, w_{k2}^0, ...w_{kj}^0, ..., w_{kD}^0\right)$ $(1 \le k \le m)$ and the historical inferior position of the i^{th} particle, denoted as $W_i = \left(w_{i1}, w_{i2}, ..., w_{ij}, ..., w_{iD}\right)$ $(1 \le i \le n)$, are used as two new exemplars to guide particle *i* flying by changing its speed and direction(whether it can be effectively applied in the adjustment of particle's position). This behavior is called reverse-learning process in this paper, which is described as follows:

$$v_{ij}^{t+1} = \omega \cdot v_{ij}^t + c_3 \cdot r_3 \cdot \left(x_{ij}^t - w_{ij}^t\right) + c_4 \cdot r_3 \cdot \left(x_{ij}^t - w_{kj}^0\right) \quad (3)$$

where *c*3 and *c*4 are the acceleration coefficients that determine the relative weight of self-cognitive and social influence from inferior positions; rand *r*3 and rand *r*4 are two random numbers that are uniformly distributed over [0, 1]; w_{kj}^0 represents the k^{th} inferior position in the j^{th} dimension; w_{ij}^t represents personal historical worst position of particle *i* in the j^{th} dimension at *t* time. The meaning of other parameters can refer to equations (1) and (2).

After reverse-learning process, some particles could jump out of local optimization and redistribute over the search space. To make these particles have a more widespread distribution condition, a longer rejection radius *R* between the inferior positions W_k^0 is necessary. So when choosing the inferior particles position in the initial population, we should account for not only the fitness of them but also the rejection radius between them. The algorithm to initialize a subpopulation consisting of *m* inferior positions that satisfy a predefined rejection radius is described as Algorithm 1.

B. Local Research by Historical Optimal Position

To obtain a accurate solution, each elite in a population should not only take responsibility for guiding the evolutionary search but also make self-directed learning. In this paper, the global historical best position *gbest* makes local research based on the differential between itself and the second global historical best position *sgbest*, which is defined as (4).

$$gbest^* = gbest + r \cdot d_t \cdot \left(gbest - sgbest\right) \quad (4)$$

where *r* is a random numbers that uniformly distributed over [0, 1], by which *gbest* could adjust direction of the local research; d_t is a weighting factor of the differential value at the t^{th} iteration. In general, the distance between *gbest* and the global optimal solution will become shorter and shorter along with evolution of population. So the radius of local research should have similar variation trend during the evolution. In this paper, d_t decreases linearly during the run time, which is updated as follow:

$$d_{t+1} = d_t \cdot \left(1 - t / T\right) \quad (5)$$

where *T* is a maximum number of search iterations and *t* is the current search iteration.

After the local research, greedy-choice strategy is applied to P_{g1} and $P_{g1}^{'}$, which is defined as (6).

$$gbest = \begin{cases} gbest^*, & if\ fit\left(gbest^*\right) > fit\left(gbest\right) \\ gbest, & other \end{cases} \quad (6)$$

where *fit*(*x*) is the fitness of position *x*.

C. Adjustable Maximum Flying Velocity $V_{\max}$

As introduced in Section II, we know that a well-designed maximum flying velocity $V_{\max}$ could make a particle search the problem space with a reasonable step length. Many experiments demonstrated that $V_{\max}$ is set in the interval of 10% ~20% of search space could obtain a preferable result. However, we think it is preferable that different $V_{\max}$ should be adopted by different particles at different stages in the optimization. For example, a particle that responsible for "exploration" requires a high flying velocity to improve probability of global search. On the contrary, and the end of optimization, a particle with stronger velocity restraint enables itself to obtain a more accurate resolution.

When the optimum information of a swarm is stagnant during the search process, what we want is to help some particles to jump out of local optimal via the reverse learning process in which a larger $V_{\max}$ could enable these particles to escape from the local optimal with a high velocity. Finishing the reverse learning, however, the particles need to adjust their velocity to a smaller value in order to meet the requirement of local search.

Together with the aforementioned components, the implementation of the RLPSO algorithm is summarized in Algorithm 2.

IV. Benchmark Tests and Discussion

A. Test Functions

The performance of the developed RLPSO is evaluated through nine well-known functions that often used as test suite in numerical experiments. The details of these functions are described as Table I. The optimal objective values of these functions for minimization are all zero.

Algorithm 1. Construction of inferior subpopulation

INPUT: Initial population $X^0 = \{X_1^0, X_2^0, ...X_N^0\}$, Rejection radius *R*, Initial inferior population $W = \{\phi\}$

OUTPUT: Population $W = \{W_1^0, W_2^0, ..., W_m^0\}$

Begin

Step 1. Evaluate X^0 and rearrange it in ascending order sort by fitness. After rearrange, without loss of generality, the population is still recorded as : $\{X_1^0, X_2^0, ..., X_N^0\}, fit_{X_i^0} \le fit_{X_j^0}, \ i, j \in [1, N] \& \& i < j$

$W_1^0 = X_1^0$, *count*=1, $W = W \cup \{W_{count}^0\}$, *i*=1;

Step 2. **FOR** *i=2* **TO** *N*

IF $count \ge m$ **Break**;

ELSEIF $\forall j\left(\| W_j^0 - X_i^0 \| > R\right), 1 \le j \le count$

count++; $W_{count}^0 = X_i^0$; $W = W \cup \{W_{count}^0\}$;

END IF

ENDFOR

Step 3. **WHILE** $count < m$

Random generate a new paticles *ind*;

IF $\forall j\left(\| ind - W_j^0 \| > R\right), 1 \le j \le count$ $count++; W_{count}^0 = ind$; $W = W \cup \{W_{count}^0\}$;

END IF

END WHILE

End

Algorithm 2. RLPSO Algorithm

INPUT: *D*-dimensional optimized function; *N* (Population size); *m* (Inferior subpopulation size); Coefficients c_1,c_2,c_3 and c_4; $\omega_{\max}$ and $\omega_{\min}$ (maximum and minimum of inertia weight); $V_{\max}$ (maximum flying velocity); L_{times}(iterations of reverse learning); *n*(the number of particles that participate in a reverse learning); d_0 (weighting factor);

OUTPUT: Population historical best position P_{g1};

Begin

Step 1. Generate initial population $X^0 = \{X_1^0, X_2^0, ...X_N^0\}$ and velocity $V^0 = (V_1^0, V_2^0, ..., V_N^0)$;Set $P_i = B_i = X_i$,*t*=0;

Step 2. Evaluate population X^t; Set P_{g1} =the best particle, P_{g2} =the second best particle;

Step 3. Construct a inferior population *W* according Algorithm 1;

Step 4. **WHILE** (Not meet the stop condition)

Step 5. Update V^t and X^t according to formulas (1) and (2), respectively;

Step 6. Evaluate population X^t; Update P_i, B_i, P_{g1} and P_{g2};

Step 7. P_{g1} performs local learning procedure according to formula (3); Update d_t according formula (4);

Step 8. **IF** Meet the reverse learning condition

Step 9. Adjust $V_{\max}$;

Step 10. Update $V_1^t \sim V_Q^t$ and $X_1^t \sim X_Q^t$ according to formula (5)and (2), respectively;

Step 11. Update $V_{Q+1}^t \sim V_N^t$ and $X_{Q+1}^t \sim X_N^t$ according to formula (1)and (2), respectively;

Step 12. **END IF**

Step 13. *t*=*t*+1;

Step 14. **END WHILE**

End

B. Experimental Setting

Three variant PSO algorithms are selected for comparisons. The first one is DEPSO algorithm [21] in which DE and PSO are adopted as two basic optimization methods. According to the current success rates of DE and PSO, DEPSO select a better one to guide subsequent optimization process. The second one is RCPSO which introduced in [18]. In RCPSO, a common single *gbest* is replaced by a group of elite particles. To keep the diversity of population, a rejection strategy is adopted by the elite particles. According to Section II, the modification of DEPSO and RCPSO can be categorized into *Hybridization strategies* and *Learning-patterns adjustment,* respectively. The last peer algorithm is a recent standard PSO algorithm named PSO2007 available on the Particle Swarm Central: http://www.particleswarm.info. For more details about these peer algorithm, the reader is referred to corresponding literatures.

In this experimental, we chose two different dimensions, including D=10 and D=100, for these test function introduced in Table I to test the scalability of algorithms on low-dimension and high-dimension function optimization. According to the different dimensions, two population sizes are set as N=20 and N=30 respectively. The configuration of each peer algorithm taken from the literature is exactly the same as that used in the original paper. Note that all optimizers involved in the experiment were implemented on a PC with 2.93GHz Pentium CPU, 4.0GB memory, Windows 7 operator system and all algorithms were implemented in MATLAB 2009a.

C. Results and Discussion

In this section, each algorithm was executed 100 independent runs over the 9 test functions in two cases, which are 10 and 100 dimensions, respectively, with the corresponding population size setting as 20 and 30, respectively. The experimental results in the form of the mean value (Mean), minimum value (Min), standard deviation value (SD) and mean run-time (RT) of finally discovered best object values are presented in Table II and Table III, where the best result of Mean and Min on each problem among all algorithms is shown in bold.

From the experimental results in Table II, it can be seen that RSPSO obtains the best mean results on F_2, F_6 and F_9 where RLPSO achieves the second-best results that slightly worse than the best results. Since F_6 and F_9 are two unimodal functions which selected to examine the exploitation ability of an algorithm, we can draw a conclusion that rejection mechanism in RSPSO and local learning mechanism in RLPSO are benefit to improve the exploitation ability and increase the solution accuracy. Meanwhile, RLPSO obtains the best results on F_1, F_3, F_7 and F_8 while SPS2007 achieves the best results on F_4 and F_5. Although DEPSO and RSPSO are much more inferior to RCPSO and RLPSO on function F_1, it should be pointed out that these four algorithms have obtained almost identical solution accuracy in most cases except DEPSO trapping in local optimum 23 out of 100 runs while SPS2007 trapping in local optimum 3 out of 100 runs. From the experimental results in Table III, we can see that, with the increasing of functional dimension from 10 to 100, RLPSO obtains the best results on most functions except on F_2 and F_6. It also should be pointed out that in spite of the mean value obtained by RLPSO on F_2 and F_6 are not the best one, the number of minimum values on F_2 that less than 10^{-14} is 79 out of 100 runs while the number of minimum values on $F6$ that less than 10^{-38} is 85 out of 100 runs. So we can say that RLPSO is the best one among the four peer algorithms especially for solving high-dimensional function when success rate and solution accuracy are considered together.

From the experimental results shown in Table II and Table III we can see that time-consuming of PSO2007 is the lowest than other algorithms benefit of its simple operators while the time-consuming of RSPSO is the longest due to its construction of elite subpopulation. For example, if the elite subpopulation size is M in RSPSO, we need to make at least M*(M-1)/2-1 Euclidean distance calculations between particles in each generation in order to maintain the elite subpopulation diversity. In addition, the rejection radius also needs to be calculated in each generation. DEPSO is more time-consuming than PSO2007 not only because of DEPSO need to make statistical analysis but also because of differential evolution algorithm is more time consuming the traditional PSO. In RLPSO, two novel learning mechanism called reverse learning and local learning are adopted. Note that there is no extra time-consuming during the reverse learning model for there is no additional operators except a reverse flying direction of some particles. Although an inferior subpopulation is needed in RLPSO, unlike the construction of elite subpopulation in RSPSO, the inferior subpopulation only constructed one time after population initialization. During the local learning process, particle P_{g1} makes local research in some different fly directions which cause the time-consuming of RLPSO is more than it of PSO2007. As the dimensions is increase from 10 to 100, P_{g1} will take more time to execute local research. So RLPSO is less time-consuming than DEPSO in 10D while RLPSO is more time-consuming than DEPSO in 100D. In fact that if we execute the local research of P_{g1} and the fly process of other particles in parallel, RLPSO can obtain the same time-consuming as PSO2007.

Aside from the aforementioned targets presented in Table 4 and Table 5, convergence speed is also an important assessment criterion for PSO algorithm. The plots in Fig 1 show the convergence progress of the mean solution values of the 50 trials during the run for functions F_4, F_7, F_8 and F_9. In the four functions, F_8 and F_9 are unimodal functions selected to test convergence speed and solving accuracy of algorithm while F_4 and F_7 are multimodal functions adopted to examine global search ability of algorithm. It can be observed from the figure that PSO2007 has faster convergence speed in solving F_8 and F_9 with low dimensions than other peer algorithms in the initial stage of evolution. However, DEPSO, RSPSO and RLPSO have faster convergence speed than PSO2007 at the later evolution process. At the same time, we see that RLPSO has the fastest convergence speed and most accurate results in solving these unimodal functions when the functional dimension increases from 10 to 100. The same conclusion can be obtained from the optimization of function F_7. During the process of optimization of function F_4 with 10 dimensions, RLPSO obtains the best result and the fastest convergence speed. However, once the dimension is 100, RLPSO only provides the second-best performance for function F_4. In fact, RLPSO obtains the best performance for all trial functions except function F_4. Due to limited space, the convergence progresses of other functions listed in Table3 are not presented in this paper. So it could be an understatement to say that RLPSO is more suitable for high-dimensional function optimization.

TABLE I.

BENCHMARK PROBLEMS

Function name	Definition	Search space	Global $f_{\min}(\boldsymbol{x}^*)$	$\boldsymbol{x}^*$
Acley	$F_1(X)=20+e-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	$[-32,32]^D$	0	0.0^D
Alpine	$F_2(X)=\sum_{i=1}^{D}\lvert x_i\sin(x_i)+0.1x_i\rvert$	$[-10,10]^D$	0	0.0^D
Rastrigin	$F_3(X)=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	$[-5.12,5.12]^D$	0	0.0^D
Schwefel P1.2	$F_4(X)=418.9829D-\sum_{i=1}^{D}\left(x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$[-500,500]^D$	0	420.9687^D
Girewank	$F_5(X)=\sum_{i=1}^{D}(x_i^2/4000)-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$	$[-600,600]^D$	0	0.0^D
Schwefel P2.22	$F_6(X)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0	0.0^D
Rosenbrock	$F_7(X)=\sum_{i=1}^{D-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right)$	$[-30,30]^D$	0	0.0^D
Sphere	$F_8(X)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0	0.0^D
Sum of different power	$F_9(X)=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$[-1,1]^D$	0	0.0^D

TABLE II.

COMPARISON RESULTS OF MEANS AND VARIANCES IN 10 DIMENSIONS (D=10, N=20)

	F_1				F_2				F_3			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	2.17e-01	3.55e-15	3.84e-01	0.83	3.02e-08	7.41e-78	5.89e-08	0.56	4.79e+00	4.46e-13	1.51e+00	0.73
RCPSO	**3.52e-15**	**0**	7.03e-17	0.92	**1.07e-15**	**6.28e-79**	1.06e-15	0.99	6.03e+00	9.95e-01	2.14e+00	0.93
SPS2007	1.16e-02	3.55e-15	2.29e-02	0.24	6.50e-06	1.70e-69	1.22e-05	0.22	5.91e+00	9.95e-01	2.02e+00	0.23
RLPSO	**3.52e-15**	**0**	7.03e-17	0.62	3.13e-15	2.08e-67	2.02e-15	0.45	**5.67e-01**	**0**	6.92e-01	0.70

	F_4				F_5				F_6			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	8.51e+02	2.38e+02	2.63e+02	0.97	7.91e-02	**0**	3.54e-02	0.97	1.52e-04	2.11e-39	2.92e-04	0.52
RCPSO	1.25e+03	2.37e+02	2.62e+02	0.93	8.24e-02	7.40e-03	3.02e-02	0.96	3.14e-23	1.70e-44	6.19e-23	0.92
SPS2007	**6.65e+02**	**1.18e+02**	1.76e+02	0.25	**3.76e-02**	**0**	2.00e-02	0.24	1.11e-31	7.39e-36	1.88e-31	0.21
RLPSO	1.08e+03	4.77e+02	2.28e+02	0.78	5.02e-02	**0**	2.66e-02	0.70	**1.05e-34**	**2.93e-45**	2.00e-34	0.55

	F_7				F_8				F_9			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	6.39e+02	2.71e-03	1.06e+03	0.52	1.81e-55	3.95e-61	2.78e-50	0.48	4.30e-103	6.03e-118	8.26e-103	1.20
RCPSO	4.88e+00	1.19e-03	4.43e+00	0.96	**1.03e-62**	**1.11e-85**	2.04e-62	0.99	**2.80e-107**	**4.00e-152**	5.50e-107	2.77
SPS2007	5.12e+00	4.82e-02	3.37e+00	0.22	1.50e-52	5.70e-61	2.88e-52	0.21	1.53e-98	3.18e-108	2.81e-98	0.68
RLPSO	**2.84e+00**	**1.15e-03**	2.36e+00	0.53	1.43e-60	3.51e-79	6.60e-60	0.55	6.75e-107	4.34e-139	1.29e-106	1.39

TABLE III.

COMPARISON RESULTS OF MEANS AND VARIANCES IN 10 DIMENSIONS (D=100, N=30)

	F_1				F_2				F_3			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	2.47e+00	1.25e+00	3.90e+00	1.87	**1.23e+00**	5.69e-02	1.15e+00	1.81	2.49e+02	1.71e+02	2.81e+01	1.99
RCPSO	2.79e+00	1.63e+00	3.96e-01	2.58	**3.03e+00**	1.72e-01	2.75e+00	2.51	2.53e+02	1.65e+02	2.99e+01	2.52
SPS2007	2.20e+00	1.22e+00	4.05e-01	1.44	8.69e+00	2.62e-01	3.70e+00	1.37	3.06e+02	1.05e+02	5.65e+01	1.44
RLPSO	**1.01e-12**	**1.81e-13**	8.53e-13	2.30	8.69e+00	**3.09e-13**	6.60e+00	2.19	**2.20e+01**	**6.96e+00**	2.44e+01	2.31

	F_4				F_5				F_6			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	1.95e+04	1.21e+04	1.81e+03	1.97	7.80e-01	2.85e-01	1.98e-01	2.03	2.96e+00	1.17e-01	2.11e+00	1.94
RCPSO	1.95e+04	**1.12e+04**	1.67e+03	2.73	1.75e+00	1.20e-01	1.79e+00	2.67	4.35e+01	3.88e-01	5.87e+01	2.09
SPS2007	**1.69e+04**	1.27e+04	1.36+03	1.53	9.16-02	1.14e-02	6.56e-02	1.54	**1.05e-01**	1.41e-02	7.57e-02	1.12
RLPSO	**1.69e+04**	1.14e+04	1.74e+03	2.31	**2.32e-02**	**0**	2.28e-02	2.16	4.50e+00	**5.89e-50**	2.55e+00	1.79

	F_7				F_8				F_9			
	Mean	Min	SD	RT	Mean	Min	SD	RT	Mean	Min	SD	RT
DEPSO	8.05e+04	4.38e+02	1.35e+05	1.74	4.62e+00	6.91e-01	2.90e+00	1.58	1.58e-25	1,53e-30	2.10e-25	1.99
RCPSO	8.11e+03	3.49e+02	1.33e+03	2.30	1.08e+02	4.37e-01	1.98e+02	2.29	6.84e-30	3.80e-35	1.17e-29	2.70
SPS2007	4.92e+02	3.01e+02	8.32e+01	1.37	2.17e-01	1.44e-01	2.44e-01	1.20	3.75e-23	7.19e-29	6.45e-23	1.51
RLPSO	**4.35e+02**	**7.21e+01**	2.99e+02	2.17	**1.99e-20**	**1.80e-25**	3.56e-20	1.80	**3.04e-51**	**1.80e-91**	5.83e-51	2.08

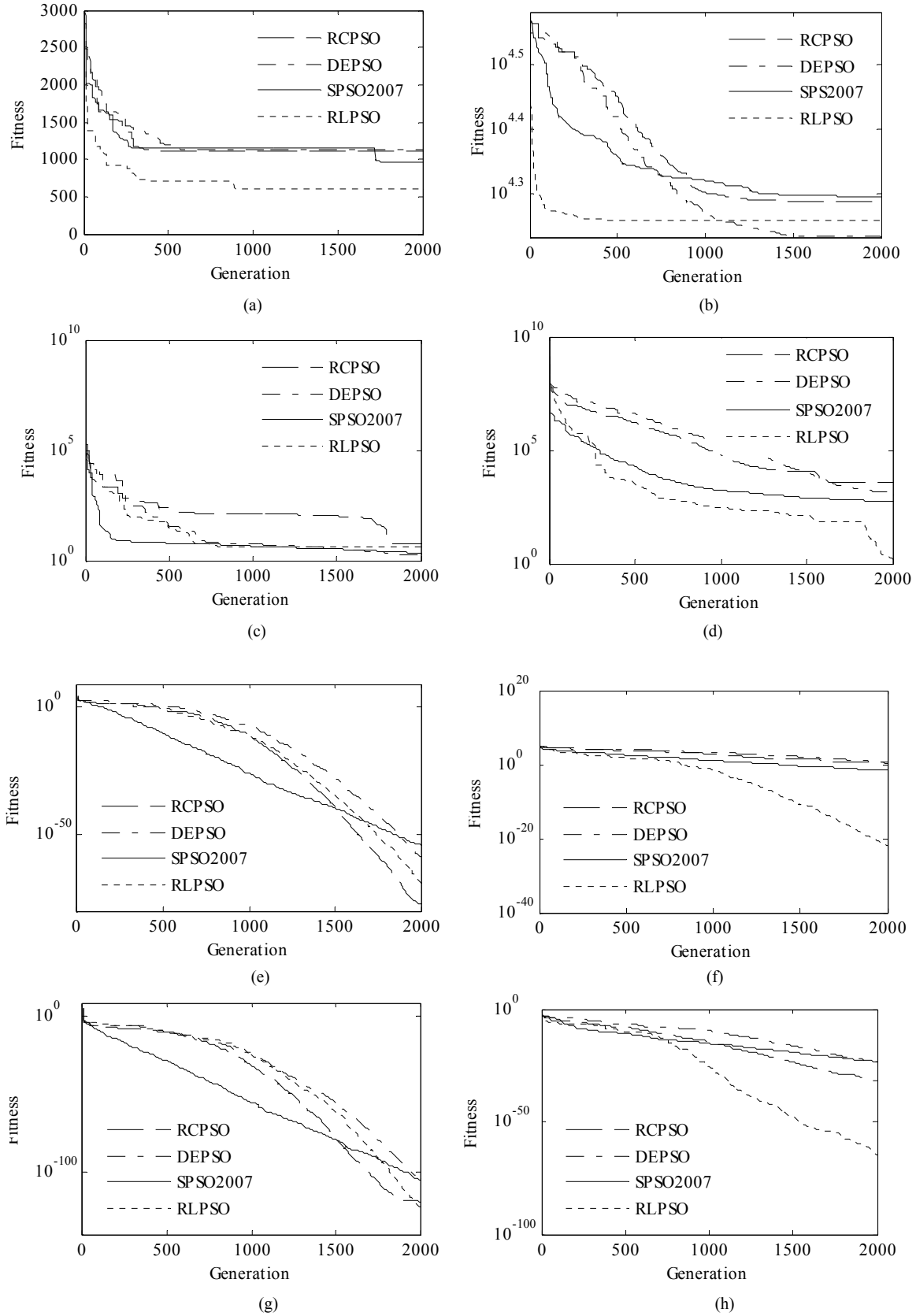

Figure 1. Comparison of convergence performance on 4 test functions in two different dimensions. (a)$F_4(10D)$. (b)$F_4(100D)$. (c)$F_7(10D)$. (d)$F_7(100D)$. (e)$F_8(10D)$. (f)$F_8(100D)$. (g)$F_9(10D)$. (h)$F_9(100D)$.

V. CONCLUSIONS

In this paper, we presented a modified PSO called RLPSO algorithm in which a reverse-learning process and a local-learning process are adopted to make the search effective and efficient. In each generation, a local-learning by the best particle is executed after standard flying process. While the population traps in local optimum, the reverse-learning process is carried out to help some particles jump out of the local optimum. During the reverse-learning process, some inferior particles in initial population and the historical inferior positions of particles are used as new exemplars to guide particles flying. While selecting the inferior particles in the initial population, we set a reject radius to make the inferior particles widely distributed in search space which can improve the diversity of population while after reverse-learning process. Furthermore, the difference of the best particle and the second-best particle in each generation is used to guide the best one to carry out local search.

Simulation results reveal that RLPSO has a good global searching ability and convergence speed especially in high dimension function.

ACKNOWLEDGMENT

The research work was jointly supported by the National Key Project for Basic Research of China (Grants No. 41231174), the National Science Fund (Grants No. 61070009) and Hubei Education Department Science Fund (Grants No. XD2012394)

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", *Proceedings of IEEE International Conference on Neural Network*, Perth, Australia, vol.4, pp.1942-1948,1995.

[2] LI Jia-sheng WANG Ying-de, TIAN wang-lan, "Extracting communication signals based on particle-swarm-optimization and adaptive-neural-fuzzy-control", *International Journal of Advancements in Computing Technology,* vol.5, no.3, pp.608-615, 2013.

[3] Baojian Zhang, Fugui Chen and Linfeng Bai, "An improve particle swarm optimization algorithm", Journal of Computer, vol. 6, no.11, pp. 2460-2467, 2011

[4] Sibin Zhu, Guixian Li and Junwei Han, "An improve PSO algorithm with object-oriented performance database for flight trajectory optimization", Journal of Computer, vol. 7, no.7, pp. 1555-1563, 2012

[5] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage*, AK, pp.69–73, 1998.

[6] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization", *Proceedings of IEEE Conference on Evolutionary Computation*, Soul, Korea, pp.101–106 , 2001

[7] FAN Hui-Lian, ZHONG Yuan-Chang, "Two-subpopulation particle swarm optimization based on pheromone diffusion", *Journal of System Simulation*, vol.23, no.10, pp.2125-2129, 2011.

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

[9] Ioan Cristian Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection". *Information Processing Letters,* vol.85, no.6, pp.317-325, 2003.

[10] M Jiang, Y P Luo, S Y Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm", *Information Processing Letters*, vol.102, no.1, pp.8-16, 2007.

[11] J. Kennedy and R. Mendes, "Population structure and particle swarm performance", *Proceedings of Congress on Evolutionary Computation*, Honolulu, HI, pp.1671–1676, 2002.

[12] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transaction on Evolutionary Computation,* vol.8, no.3, pp. 204–210, 2004.

[13] Suganthan P.N., "Particle swarm optimiser with neighborhood operator", *Proceedings of IEEE Congress on Evolutionary Computation*, Washington, D.C., USA, pp.1958–1962, 1999.

[14] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator", *Proceedings of IEEE Congress on Evolutionary Computation*, pp.1958–1962, 1999.

[15] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Transaction on Systems, Man, and Cybernetics – Part B, Cybernetics*, vol.35, no.6, pp. 1272–1282, 2005.

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

[17] Changhe Li, Shengxiang Yang, "A Self-learning particle swarm optimizer for global optimization problems", *IEEE Transaction on Systems, Man, and Cybernetics – Part B, Cybernetics*, vol.42, no.3, pp. 627–646, 2012.

[18] TAN Yang, TANG De-Quan, QUAN Hui-Yun, "Rejection of competition particle swarm optimization", *Journal of System Simulation*, vol.23, no.2, pp. 2635-2640, 2011.

[19] Ji Qiang Zhai, Ke Qi Wang, "Baldwin effect based particle swarm optimizer for multimodal optimization", Journal of Computer, vol. 7, no.9, pp. 2114-2119, 2012

Xuewen Xia, born in 1974, Ph. D., assistant professor. He received PhD from Wuhan University in 2009. His research interests include computation intelligence and application.

Jingnan Liu, born in 1943, professor, Ph.D. supervisor, academician of Chinese Academy of Engineering. His research interests include GPS theories, method and data processing.

Yuanxiang Li, born in 1962, Ph.D., professor, Ph.D. supervisor. He received PhD from Wuhan University in 1993. His research interests include parallel computation, computation intelligence and application.