Open access • Proceedings Article • DOI:10.1109/CEC.2009.4982972

An adaptive learning particle swarm optimizer for function optimization

Changhe Li, Shengxiang Yang

Institutions: University of Leicester

Published on: 18 May 2009 - Congress on Evolutionary Computation

Topics: Particle swarm optimization, Premature convergence, Adaptive learning and Local optimum

An Adaptive Learning Particle Swarm Optimizer for Function Optimization

Changhe Li and Shengxiang Yang

***Abstract*— Traditional particle swarm optimization (PSO) suffers from the premature convergence problem, which usually results in PSO being trapped in local optima. This paper presents an adaptive learning PSO (ALPSO) based on a variant PSO learning strategy. In ALPSO, the learning mechanism of each particle is separated into three parts: its own historical best position, the closest neighbor and the global best one. By using this individual level adaptive technique, a particle can well guide its behavior of exploration and exploitation. A set of 21 test functions were used including un-rotated, rotated and composition functions to test the performance of ALPSO. From the comparison results over several variant PSO algorithms, ALPSO shows an outstanding performance on most test functions, especially the fast convergence characteristic.**

I. Introduction

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in [1], [2]. PSO is motivated from the social behavior of organisms, such as bird flocking and fish schooling. In PSO, a swarm of particles "fly" through the search space. Each particle follows the previous best position found by its neighbor particles and the previous best position found by itself. In the past decade, PSO has been actively studied and applied for many academic and real world problems with promising results due to its property of fast convergence [8].

Ever since PSO was first introduced, several major versions of the PSO algorithms have been developed [8]. Each particle is represented by a position and a velocity, which are updated as follows:

$$V'^{d}_{i} = \omega V^{d}_{i} + \eta_1 r_1 (pbest^{d}_{i} - X^{d}_{i}) + \eta_2 r_2 (gbest^{d} - X^{d}_{i}) \quad (1)$$

$$X'^{d}_{i} = X^{d}_{i} + V'^{d}_{i}, \quad (2)$$

where X'^{d}_{i} and X^{d}_{i} represent the current and previous position of $d-th$ dimension of particle i respectively, V'_i and V_i are the current and previous velocity of particle i respectively, $pbest_i$ and $gbest$ are the best position found by particle i so far and the best position found by the whole swarm so far respectively, $\omega \in (0, 1)$ is an inertia weight, which determines how much the previous velocity is preserved, η_1 and η_2 are the acceleration constants, and r_1 and r_2 are random numbers generated in the interval $[0.0, 1.0]$.

There are two main models of the PSO algorithms, called $gbest$ (global best) and $lbest$ (local best), which differ in the way of defining the neighborhood of each particle. In the $gbest$ model, the neighborhood of a particle consists of the particles in the whole swarm, which share information between each other. On the contrary, in the $lbest$ model, the neighborhood of a particle is defined by several fixed particles. The two models give different optimization performances on different problems. Kennedy and Eberhart [3] and Poli et al. [8] pointed out that the $gbest$ model has a faster convergence speed with a higher chance of getting stuck in local optima than $lbest$. On the contrary, the $lbest$ model is less vulnerable to the attraction of local optima but with a slower convergence speed than the $gbest$ model.

In order to improve PSO's performance, we present an adaptive learning PSO (ALPSO) that utilizes a new learning strategy. In ALPSO, each particle can adjust its search strategy according to the selection ratios of four learning operators in different surrounding environments. The selection ratio of each operator is calculated in the same way as in [4]. For the global best particle, we introduce a learning method that can subtract the promising information from all improved particles.

The rest of this paper is organized as follows. Section II describes the adaptive learning PSO. The experimental study is present in section III and finally conclusions are given in section IV.

II. Adaptive Learning Particle Swarm Optimizer

Although there are many improved versions of PSO, how to balance the performance of the $gbest$ and $lbest$ models is still an important issue, especially for multi-modal problems. In the $gbest$ model, all particles' social behavior is strictly constrained by learning information from the global best particle. Hence, particles are easily attracted by $gbest$ and quickly converge on that region even it is not the global optimum and $gbest$ does not improve. In the $lbest$ model, attraction by the $gbest$ is not too much but the slow convergence speed is unbearable. In the origin PSO, each particle learns from its $pbest$ and the $gbest$ simultaneously, which might cause the above problems. Hence, we can separate the cognition component and the social component to increase diversity, but the proper moment for a particle to learn from $gbest$ or $pbest$ is very hard to know. The following sections will give an adaptive method to enable a particle to automatically learn from the global or local information from different particles.

A. Learning Strategy in ALPSO

In ALPSO, the information learnt by each particle comes from four sources: the $gbest$, its own $pbest$, the $pbest$ of

The authors are with the Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, United Kingdom (email: {cl160, s.yang}@mcs.le.ac.uk)

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom under Grant EP/E060722/1.

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

the closest particle, and a random position around itself. The learning equations are as follows:

$$a : V_i^d = \omega V_i^d + \eta \cdot r_i^d \cdot (pbest_i^d - X_i^d) \quad (3)$$

$$b : V_i^d = \omega V_i^d + \eta \cdot r_i^d \cdot (pbest_{i\_nearest}^d - X_i^d) \quad (4)$$

$$c : X_i^d = X_i^d + V_{avg}^d \cdot N(0,1) \quad (5)$$

$$d : V_i^d = \omega V_i^d + \eta \cdot r_i^d \cdot (gbest^d - X_i^d) \quad (6)$$

where $pbest_{i\_nearest}$ is the $pbest$ of the closest particle to particle i, V_{avg} is the average velocity of all particles, and $N(0,1)$ is a random number from the normal distribution with mean 0 and variance 1.

Learning from the nearest neighbor enables a particle to explore the region of local optima around itself. Particles that are near a local optimum will get closer and closer to that region because the $pbest$ is replaced only when a better position is found. Gradually, they will generate a local cluster around that local optimum. Particles in one local cluster are not influenced by those far away (other local clusters) even they have very good fitness. This strategy can help swarm find more local optima rather than one optimum as the original PSO does, especially for multi-modal problems.

Once particles converge on a local optimum or there is a more promising region nearby without particles covering it, particles should have a probability to jump to that promising region. Hence, learning from a random position around itself is needed.

In ALPSO, each particle has four different choices to adjust its behavior. The four choices enable each particle to move to a promising position with a higher probability than the original PSO. Here, which choice is the most suitable depends on the around environment where a particle is. However, we can not know what the around environment looks like. Each particle should detect the shape of the environment where it is by itself. Hence, we use the method proposed in [4], which enables a particle to choose the most suitable operator automatically. The method is described in the following section.

B. The Adaptive Learning Mechanism

Borrowed the idea of probability matching[12], we introduce an adaptive framework using the aforementioned four learning operators, each of which is assigned a selection ratio. The selection ratio of each operator is equally initialized to $1/4$ and is adaptively updated according to its relative performance.

For each particle, one of the four learning operators is selected according to their selection ratios and its offspring fitness is evaluated. The operator that results in higher fitness values of offspring will have its selection ratio increased. The operator that results in lower fitness values of offspring will have its selection ratio decreased. Gradually, the most suitable operator will be chosen automatically and control the leaning behavior of each particle in different environments.

Without lose of generality, we discuss the minimization optimization problems in this paper. Based on our previous work in [4], we extend the adaptive framework at the population level into the individual level in this paper. The selection ratios are updated every U_f generations, where U_f is called the updating frequency. During the updating period for each particle, the progress value and the reward value of operator i are calculated as follows.

The progress value $prog_i(t)$ of operator i at generation t is defined as:

$$prog_i(t) = \sum_{j=1}^{M_i} f(p_j^i(t)) - \min\left(f(p_j^i(t)), f(c_j^i(t))\right), \quad (7)$$

where $p_j^i(t)$ and $c_j^i(t)$ denote a particle and its child produced by operator i at generation t and M_i is the selection times of operator i with the particle.

The reward value $reward_i(t)$ of operator i at generation t is defined as follows:

$$\begin{aligned} reward_i(t) \quad = \quad & exp(\tfrac{prog_i(t)}{\sum_{j=1}^{N} prog_j(t)}\alpha + \tfrac{s_i}{M_i}(1-\alpha)) \\ & + c_i p_i(t) - 1 \end{aligned} \quad (8)$$

where s_i is the counter that records the number of children that are fitter than their parent particles by applying operator i, $p_i(t)$ is the selection ratio of operator i at generation t, α is a random weight between 0.0 and 1.0, N is the number of operators, and c_i is a penalty factor for operator i, which is defined as follows:

$$c_i = \begin{cases} 0.9, & \text{if } s_i = 0 \text{ and } p_i(t) = \max_{j=1}^{N}(p_j(t)) \\ 1, & \text{otherwise} \end{cases} \quad (9)$$

With the above definitions, the selection ratio of operator i is updated every U_f generation according to the following equation:

$$p_i(t+1) = \frac{reward_i(t)}{\sum_{j=1}^{N} reward_j(t)}(1 - N * \gamma) + \gamma, \quad (10)$$

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

C. Information Learning for gbest

In the original PSO, the $gbest$ is updated only when particles find a better position than the current $gbest$. Once it is updated, the information of all dimensions of the $gbest$ is replaced with that of the better position. This updating mechanism has a disadvantage that promising information of some dimensions of one particle can not be kept due to bad information in other dimensions that cause its low fitness. This problem is called "Two step forward, one step back" in [13]. If a particle gets better, information of some dimension probably becomes more promising. Other particles should learn some useful information from the improved one even the particle's fitness is very low. In ALPSO, the $gbest$ learns the useful information from those dimensions of a particle that is improved. Once promising information is extract from those improved dimensions of that particle, the information of corresponding dimensions of the $gbest$ is updated. The updating happens only when particles are improved, which is as shown in Algorithm 1.

Algorithm 1 GbestUpdate(particle p)

for each dimension d of $gbest$ **do**
  $X_{t\_gbest} := X_{gbest}$, $X_{t\_gbest}[d] := X_p[d]$
  if t_gbest is better than $gbest$ **then**
    $X_{gbest}[d] := X_{t\_gbest}[d]$
  end if
end for

Algorithm 2 The ALPSO Algorithm

Generate the initial particles by randomly generating the position and velocity for each particle
Set the generation counter $t := 0$
while the stop criterion is not satisfied **do**
  for each particle i **do**
    Select one learning operator according to its selection ratio to update particle i
    if the updated particle i is better than its $pbest$ **then**
      Update $pbest$
      Perform $GbestUpdate(i)$ for $gbest$
    end if
    if the updated particle i is better than $gbest$ **then**
      Update $gbest$
    end if
    if $t\%U_f == 0$ **then**
      Update the selection ratio for each learning operator according to Eq. (10)
    else
      Calculate the accumulative reward value of each operator
    end if
  end for
  $t := t + 1$
end while

We can not apply this strategy to all particles because the learning method is time comsuming. Hence, we choose the $gbest$ as the learner. The framework of the ALPSO algorithm is given in Algorithm 2.

III. Experimental Study

A. Test Functions

In order to test the performance of ALPSO, we choose three unimodal functions and 18 multimodal functions, which are widely used as the test functions in the literature [5], [11], [14]. The details of these test functions are given in Table I. Function f_{16} is a composition function proposed by Jiang et al. [5], which is composed of ten benchmark functions: two rotated and shifted f_1, f_2, f_3, f_4, and f_5. Functions f_{18} to f_{21} are rotated functions, where the rotation matrix $\vec{M}$ for each function is obtained using the method in [9].

B. Experimental Setting

Experiments were conducted to compare five PSO algorithms on the 21 test problems. The algorithms are listed as follows:

- Standard PSO;
- CPSO-H$_k$ [13];
- FIPS [7];
- CLPSO [6];
- ALPSO

For the standard PSO, the acceleration constants η_1 and η_2 are both set to be 1.49618 and the inertia weight $\omega = 0.729844$. Equations 1 and 2 are used for the velocity and position update in the standard PSO. CPSO-H$_k$ [13] is a cooperative PSO model combined with standard PSO, the same value of $k = 6$ in [13] is used. The fully informed PSO (FIPS) [7] with a U-ring topology that achieved the highest success rate is used. Comprehensive learning PSO (CLPSO) [6] uses all other particles' historical best information to update a particle's velocity. CLSPO is designed for solving multimodal problems, and it presents a good performance in [6] compared with eight other PSO algorithms. To achieve better performance of ALPSO, we use particular settings by experience for each problem due to different complexity of different problems. In ALPSO, parameters are the same as standard PSO and the updating frequency is present in Table II. Each problem with 10 dimensions was independently run 30 times. The initial population is the same for all algorithms on each test problem and the population size is given in Table II. The maximal number of fitness evaluations is set to 100000 for all algorithms on each test problem. The code of the five algorithms is available online at the following website:
`www.cs.le.ac.uk/people/cl160/ALPSO.rar`.

C. Experimental Results and Discussions

1) Experimental Results: Table III presents the results of mean and variance values over 30 runs for the five algorithms on all test problems. The best results of each problem are shown in bold except function f_6, on which all five algorithms obtained the global optimum 0. Two-tailed T-test with 58 degrees of freedom at a 0.05 level of significance was conducted between ALPSO and the best results obtained by one of the other four algorithms and the results are also shown in Table III, where "***" means the result of two algorithms is the same. The performance difference is significant if the absolute value of the T-test result is greater than 1.984. Figs. 1 and 2 describe the convergence speed of the five PSOs on all test problems.

Form Table III, ALPSO shows an outstanding performance on functions f_1, f_8, f_{12}, and f_{13} over the other four algorithms. Especially for functions f_2, f_3, f_6, and f_9, ALPSO obtained the global optimum over all 30 runs. Comparing ALPSO with CLPSO, though the performance of ALPSO is worse than that of CLPSO on functions f_4, f_{16}, f_{20}, and f_{21}, it is much better than that of CLPSO on functions f_4, f_{12}, f_{13}, and f_{17}, and is similar to or the same as that of CLPSO on the other functions.

For unimodal functions, ALSPO shows a fast convergence speed to the global optima. For multimodal functions, ALPSO and CLPSO present a much better performance than

TABLE I

THE TEST FUNCTIONS, WHERE n AND f_{min} ARE THE NUMBER OF DIMENSIONS AND THE MINIMUM VALUE OF A FUNCTION RESPECTIVELY AND $S \in R_n$

Test Function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	10	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	10	[-5.12, 5.12]	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20$	10	[-0.5,0.5]	0
$f_4(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	10	[-600, 600]	0
$f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	10	[-32, 32]	0
$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	10	[-100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + U(0, 1)$	10	[-1.28, 1.28]	0
$f_8(x) = \sum_{i=1}^{n} 100(x_{i+1}^2 - x_i)^2 + (x_i - 1)^2)$	10	[-30, 30]	0
$f_9(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	10	[-500, 500]	-4189.829
$f_{10}(x) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	10	[-500, 500]	0
$f_{11}(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[-10, 10]	0
$f_{12}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	10	[-100, 100]	0
$f_{13}(x) = \max_{i=1}^{n} \lvert x_i \rvert$	10	[-100, 100]	0
$f_{14}(x) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4), y_i = 1 + (x_i + 1)/4$	10	[-50, 50]	0
$f_{15}(x) = 0.1\{10\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	10	[-50, 50]	0
$f_{16}(x) =$ Composition function 5 (CF5) in [5]	10	[-5, 5]	0
$f_{17}(x) = \sum_{i=1}^{n} 100(y_{i+1}^2 - y_i)^2 + (y_i - 1)^2), \vec{y} = \vec{M} * \vec{x}$	10	[-100, 100]	0
$f_{18}(x) = \frac{1}{4000} \sum_{i=1}^{n} (y_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{y_i - 100}{\sqrt{i}}) + 1, \vec{y} = \vec{M} * \vec{x}$	10	[-600, 600]	0
$f_{19}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi y_i)) + 20 + e$, $\vec{y} = \vec{M} * \vec{x}$	10	[-32, 32]	0
$f_{20}(x) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10), \vec{y} = \vec{M} * \vec{x}$	10	[-5, 5]	0
$f_{21}(x) = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (y_i + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20, \vec{y} = \vec{M} * \vec{x}$	10	[-0.5,0.5]	0

TABLE II

POPULATION SIZE AND UPDATING FREQUENCY, WHERE THE UPDATING FREQUENCY IN SHOWN IN THE BRACKET

f_1	f_2	f_3	f_4	f_5	f_6	f_7
5(5)	10(5)	10(5)	20(5)	10(5)	10(5)	20(5)
f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
10(1)	40(5)	40(5)	10(5)	5(5)	5(5)	10(5)
f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}
10(5)	40(10)	10(5)	20(5)	30(10)	20(5)	20(15)

the other three algorithms. For example, on Schwefel's functions f_9 and f_{10}, all the other three algorithms are trapped into local optimum that are far away from the glbal optimum. However, ALPSO and CLPSO both successfully avoid falling into the deep local optimum. For rotated functions, the two algorithms also show a leading performance compared with the other three algorithms.

Among the other three algorithms, CPSO-H_6 presents a comparatively better performance on most problems, and it obtains the best results on problem f_7, which is a little better than the results got by ALPSO. The FIPS with the U-ring model gives relatively better results compared with the standard PSO. The standard PSO falls into local optima on almost all multimodal problems.

From Figs. 1 and 2, one interesting observation is that ALPSO presents the fastest convergence speed on all test problems. The results obviously show that the learning strategy for $gbest$ is efficient to solve the "Two step forward, one

TABLE III
COMPARISON RESULTS OF MEANS AND VARIANCES

Function	f_1	f_2	f_3	f_4	f_5	f_6	f_7
ALPSO	**9.42e-163**	**0**	**0**	0.0405	6.25e-15	0	0.000655
	(±4.97e-162)	(±0)	(±0)	(±0.0298)	(±2.55e-15)	(±0)	(±0.000353)
CLSPO	3.81e-155	**0**	**0**	**0.00411**	**4.12e-15**	0	0.00106
	(±2.09e-154)	(±0)	(±0)	(±0.00615)	(±6.49e-16)	(±0)	(±0.000422)
FIPS	5.16e-053	11.3	**0**	0.476	0.599	0	0.00599
	(±2.83e-052)	(±5.54)	(±0)	(±0.0884)	(±3.28)	(±0)	(±0.00201)
CPSO-H_6	1.23e-103	0.133	7.11e-16	0.0343	1.02e-14	0	**0.000509**
	(±6.74e-103)	(±0.344)	(±2.17e-15)	(±0.0166)	(±4.27e-15)	(±0)	(±0.000305)
PSO	2.79e-027	17.7	1	0.0932	0.888	0	0.000891
	(±1.53e-026)	(±10.8)	(±0.88)	(±0.0505)	(±1.24)	(±0)	(±0.000648)
T-test	-1	***	***	6.54293	4.43362	***	1.71779
Function	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
ALPSO	**0.173**	**-4.19e+03**	**0.000127**	7.05e-51	**3.81e-51**	**3.15e-68**	**1.57e-32**
	(±0.617)	(±9.25e-13)	(±2.76e-20)	(±3.6e-50)	(±2.08e-50)	(±1.73e-67)	(±5.57e-48)
CLSPO	0.934	**-4.19e+03**	**0.000127**	**5.05e-52**	3.2e-06	0.0371	**1.57e-32**
	(±1.68)	(±9.25e-13)	(±2.76e-20)	(±1.42e-51)	(±1.14e-05)	(±0.748)	(±5.57e-48)
FIPS	8.09	-3.7e+03	488	3.47e-17	836	7.91e-14	5e-26
	(±6.92)	(±162)	(±162)	(±4.25e-17)	(±1.9e+03)	(±1.38e-13)	(±1.82e-25)
CPSO-H_6	5.42	-3.83e+03	355	1.38e-45	3.46e-14	5.51e-19	**1.57e-32**
	(±6.87)	(±189)	(±189)	(±7.54e-45)	(±1.9e-13)	(±2.44e-18)	(±5.57e-48)
PSO	9.32e+03	-3.35e+03	841	1	3.67e+03	4.82e-045	2.82
	(±2.74e+04)	(±305)	(±305)	(±3.05)	(±6.07e+03)	(±2.55e-44)	(±6.36)
T-test	-1.865	***	***	0.9954	-1.0	-1.03401	***
Function	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}
ALPSO	**1.35e-032**	120	**2.63**	0.101	**1.75e-014**	4.97	0.66
	(±0)	(±169)	(±6.31)	(±0.0458)	(±7.19e-015)	(±2.98)	(±0.735)
CLSPO	**1.35e-032**	**86.8**	4.07	**0.0175**	1.51e-013	**0.763**	**8.3e-006**
	(±0)	(±36.1)	(±3.35)	(±0.0111)	(±4e-013)	(±0.675)	(±1.68e-05)
FIPS	8.67	347	1.33e+07	0.522	0.00029	19.7	0.00129
	(±47.5)	(±179)	(±7.28e+07)	(±0.1)	(±0.000126)	(±4.24)	(±0.000636)
CPSO-H_6	**1.35e-32**	593	121	0.136	0.27	7.33	2.22
	(±0)	(±398)	(±176)	(±0.0676)	(±0.497)	(±3.91)	(±1.78)
PSO	81.6	848	2.44e+5	0.139	0.0385	11.6	2.14
	(±118)	(±195)	(±4.94e+5)	(±0.0648)	(±0.211)	(±5.37)	(±1.68)
T-test	***	1.05106	-1	9.65691	-1	7.55182	4.92042

step back" problem. It does help the $gbest$ learn promising information from those improved particles.

Fig 3 presents the selection ratio of each operator for some test problems. From the results, we can see that the selection ratio of each operator is quite different from problem to problem. Even for a particular test problem, the selection ratio of the best operator changes in different evolving stages.It can be seen from the results of F_2, F_3, F_5 and F_6 in Fig. 3, most particles quickly learn from the best particle when the run starts, however, the selection ratio of learning from particles' private best position surpasses the selection ratio of learning from the best particle when the number of generations reaches 500. The results validate our prediction that different learning strategies are needed in different evolving stages.

2) Discussions: From the above results on the 21 test problems, we can conclude that ALPSO performs much better than the other three algorithms on all unimodal test problems due to the learning strategy for $gbest$. Though ALPSO does not perform the best on all multimodal test problems, it presents competitive results compared with the other three improved PSO algorithms. We can also conclude that the adaptive learning mechanism enables particles to have more chances to move to a more promising region, especially for those particles being trapped into local optima in multimodal problems.

IV. CONCLUSIONS

This paper presents an adaptive learning PSO which uses an adaptive framework on the individual level to adapt four leaning strategies for each particle in the swarm. A new

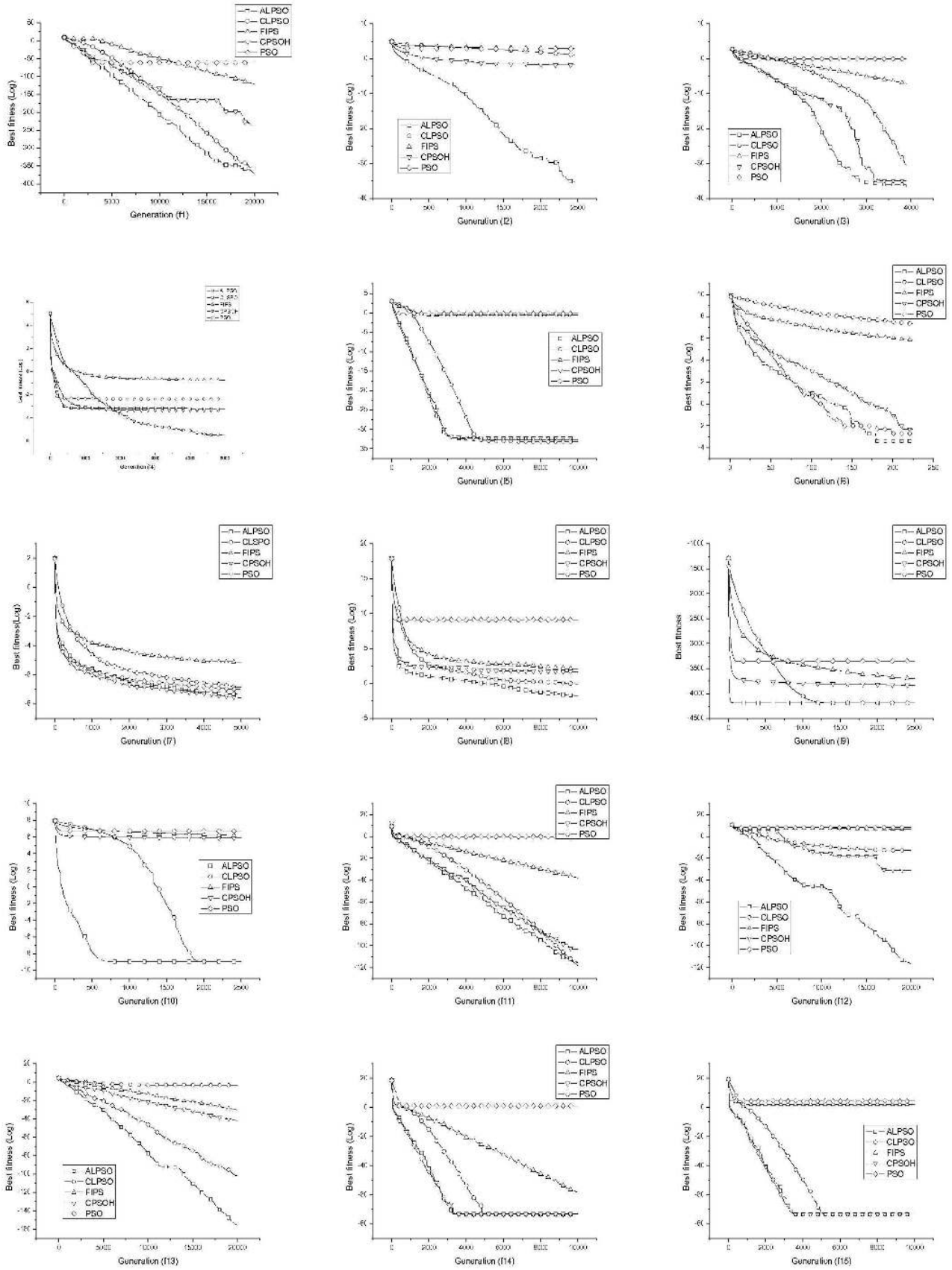

Fig. 1. Evolution process of the average best fitness of PSO, CLPSO, CPSO-H$_6$, FIPS, and ALPSO on functions f_1 to f_{15}.

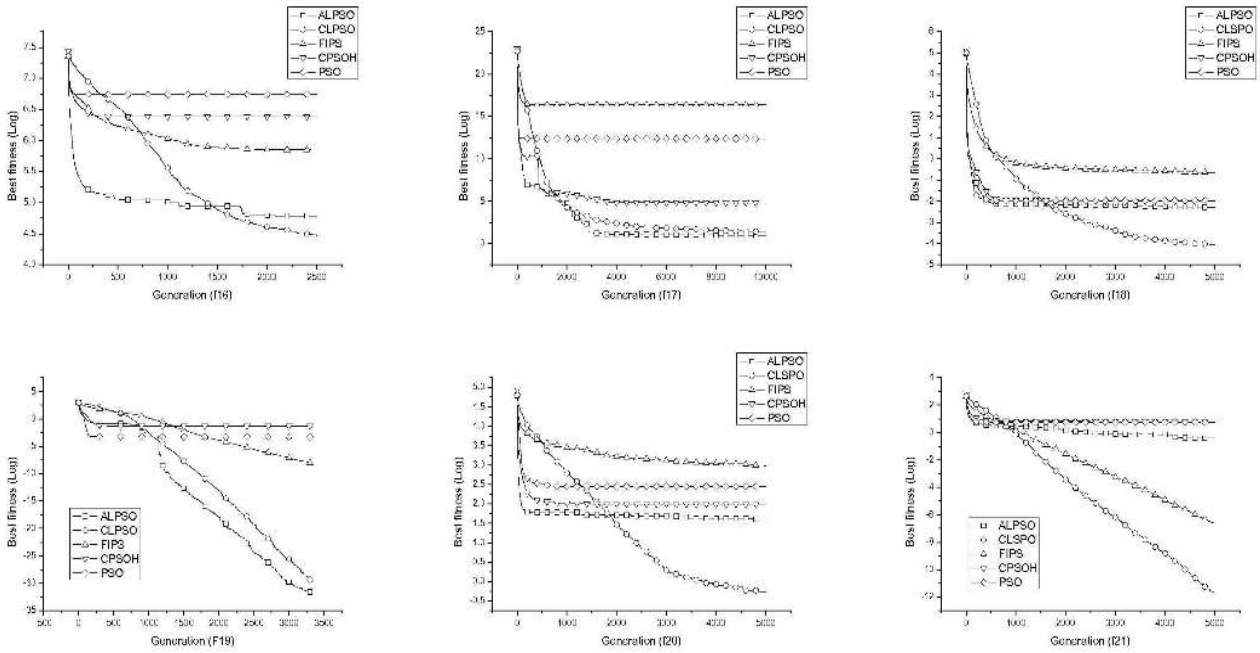

Fig. 2. Evolution process of the average best fitness of PSO, CLPSO, CPSO-H$_6$, FIPS, and ALPSO on functions f_{16} to f_{21}.

learning mechanism for $gbest$ is introduced by extracting useful information from those improved particles on all dimensions.

The four learning strategies give each particle more chances to search a larger space. A particle is not simply influenced by its own previous best position and the global best one, the nearest neighbor also can help it search in a local region. The balance of local search and the global search can be solved using the adaptive technique, which enables each particle make its own choice according to the environment around. The performance of ALPSO is tested on 21 test problems in comparison with other three improved PSOs and the standard PSO. The results show that ALPSO gives the best performance on all test unimodal problems and also presents the outstanding performance on most multimodal problems.

Although ALPSO is not the best one for solving all test multimodal problems, the adaptive framework can help particles decide their own step direction. Especially for real problems, we can not know the distribution of solution space. However, we can design different strategies to deal with different situations, and let particles choose the most suitable strategy by themselves according to the adaptive technique.

REFERENCES

[1] Eberhart, R. C. and J. Kennedy. A new optimizer using particle swarm theory. *Proc. of the 6th Int. Symp. on Micro Machine and Human Science*, pp. 39-43, 1995.

[2] Kennedy, J. and R. C. Eberhart. Particle Swarm Optimization. *Proc. of the 1995 IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995.

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

[4] C. Li, S. Yang, and I. A. Korejo. An adaptive mutation operator for particle swarm optimization. *Proc. of the 2008 UK Workshop on Computational Intelligence*, pp. 165-170, 2008.

[5] J.J. Liang, PN. Suganthan, and K. Deb. Novel composition test functions for numerical global optimization. *Swarm Intelligence Symposium, 2005*. pp. 68-75, 2005.

[6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. on Evol. Comput.*, **10**(3): 281-295, 2006.

[7] R. Mendes, J. Kennedy, and J. Neves, The fully informed particle swarm: simple, maybe better.*IEEE Trans. on Evol. Comput.*, **8**(3):204-210, 2004.

[8] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence* , **1**(1): 33-58, 2007.

[9] R. Salomon. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, **39**(3): 263-278, 1996.

[10] Shi, Y. and R. C. Eberhart. A modified particle swarm optimizer. *Proc. of the IEEE Int. Conf. on Evol. Comput.*, pp. 69-73, 1998.

[11] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *Technical Report*, Nanyang Technological University, Singapore, 2005.

[12] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities.*Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 1539-1546, New York, NY, USA, 2005.

[13] F. van den Bergh, A. P. Engelbrecht. A Cooperative approach to particle swarm optimization. *IEEE Trans. on Evol. Comput.*, **8**(2): 225-239, 2004.

[14] X. Yao, Y. Liu and G. Lin. Evolutionary programming made faster. *IEEE Trans. on Evol. Comput.*, **3**(2): 82-102, 1999.

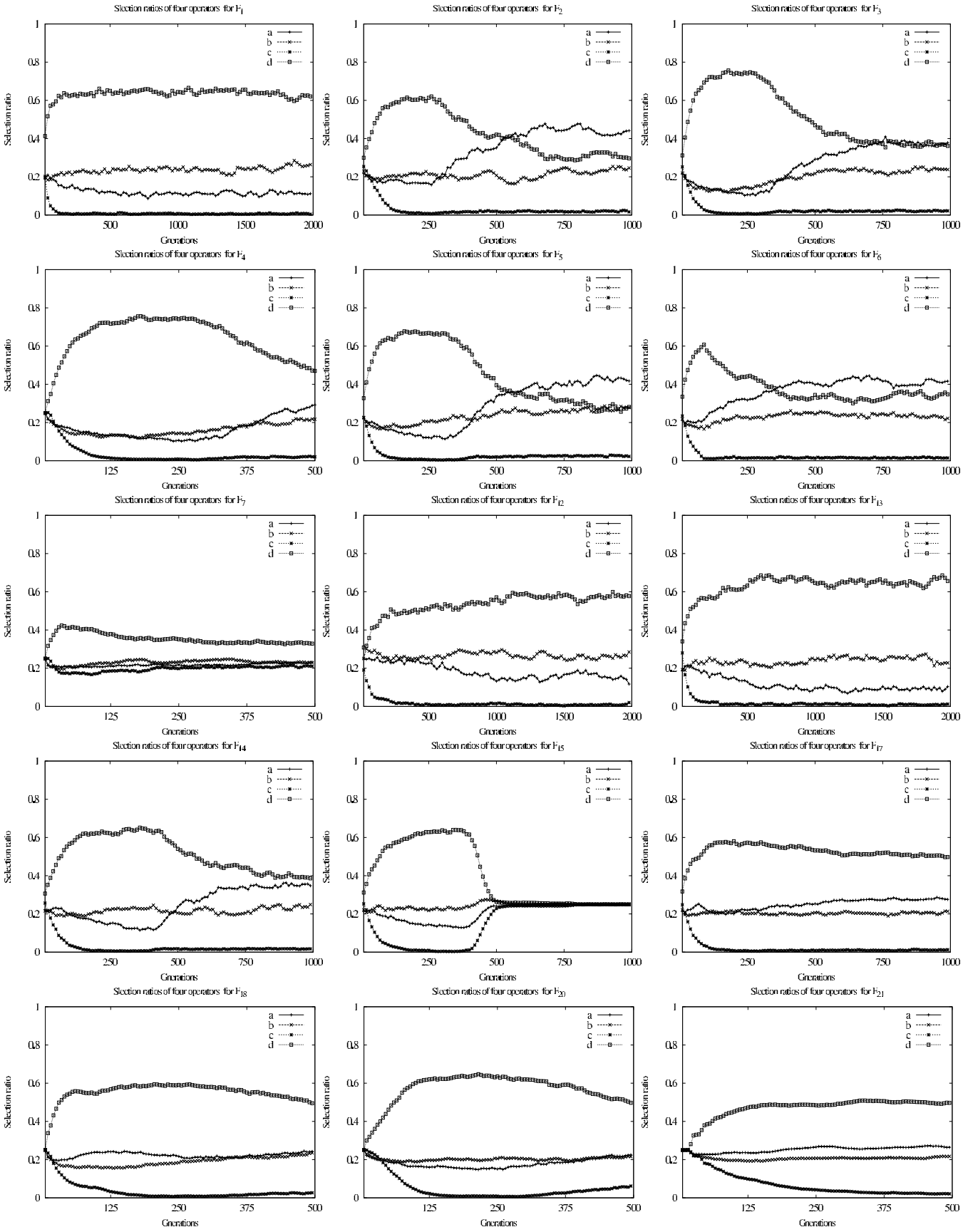

Fig. 3. The process of selection ratio of each operator for different problems