Adaptive Learning Particle Swarm Optimizer-II for Global Optimization

Changhe Li and Shengxiang Yang *Member, IEEE*

***Abstract*— This paper presents an updated version of the adaptive learning particle swarm optimizer (ALPSO) [6], we call it ALPSO-II. In order to improve the performance of ALPSO on multi-modal problems, we introduce several new major features in ALPSO-II: (i) Adding particle's status monitoring mechanism, (ii) controlling the number of particles that learn from the global best position, and (iii) updating two of the four learning operators used in ALPSO. To test the performance of ALPSO-II, we choose a set of 27 test problems, including un-rotated, shifted, rotated, rotated shifted, and composition functions in comparison of the ALPSO algorithm as well as several state-of-the-art variant PSO algorithms. The experimental results show that ALPSO-II has a great improvement of the ALPSO algorithm, it also outperforms the other peer algorithms on most test problems in terms of both the convergence speed and solution accuracy.**

I. Introduction

Particle swarm optimization (PSO) is an effective optimization tool, especially for solving global optimization problems. Since PSO was first proposed in 1995 [1], [4], it has been widely studied due to its effectiveness and simpleness. However, many experiments have shown that the basic PSO algorithm easily falls into local optima when solving complex multi-modal problems [3] and it is difficult for PSO to jump out of that local optimum once it is trapped in a local optimum.

In the literature of PSO, maintaining diversity, population topology, hybridization with auxiliary search operators and adaptive PSO have become four of the most promising approaches to preventing PSO from being trapped in local optima. In order to accelerate the convergence speed and avoid PSO from being trapped in local optima, an ALPSO [6] algorithm was proposed based on four learning operators. In order to enable particles to automatically choose the appropriate learning operator at the appropriate moment during the search process, an adaptive selection mechanism is introduced in ALPSO.

In this paper, we present the ALPSO-II, which is an updated version of the ALPSO algorithm [6]. In ALPSO-II, we introduce several new functions to improve the performance of ALPSO on multi-modal problems. These new functions include two updated learning operators, particle status monitoring mechanism, and controlling the number of particles that learn from the global best position. These approaches can increase population diversity, as a result, ALPSO-II has a higher probability of exploring more promising areas in the fitness landscape than the ALPSO algorithm.

The rest of this paper is organized as follows. Section II gives an introduction of the ALPSO algorithm. The new features of ALPSO-II algorithm are described in section III. Experimental study and results are present in section IV. Finally, conclusions and future work are discussed in section V.

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

C. Li and S. Yang are with the Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, U. K. (email: {cl160, s.yang}@mcs.le.ac.uk).

II. Adaptive Learning Particle Swarm Optimizer

In the basic PSO, each particle i is represented by a position vector $\vec{x}_i$ and a velocity vector $\vec{v}_i$, which are updated as follows:

$$v'^d_i = \omega v^d_i + \eta_1 r_1 (x^d_{pbest_i} - x^d_i) + \eta_2 r_2 (x^d_{gbest} - x^d_i) \quad (1)$$

$$x'^d_i = x^d_i + v'^d_i, \quad (2)$$

where x'^d_i and x^d_i represent the current and previous positions in the d-th dimension of particle i respectively; v'_i and v_i are the current and previous velocity of particle i respectively; $\vec{x}_{pbest_i}$ and $\vec{x}_{gbest}$ 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 $[0.0, 1.0]$ uniformly.

There are two main models of the PSO algorithm, called $gbest$ (global best) and $lbest$ (local best), respectively. The two models give different performances on different problems. Generally speaking, people believe that the $gbest$ model has a faster convergence speed but also has a higher probability of getting stuck in local optima than the $lbest$ model [5], [8]. On the contrary, the $lbest$ model is less vulnerable to the attraction of local optima but has a slower convergence speed than the $gbest$ model.

In order to alleviate the problems of the two models and enhance the advantages of them, ALPSO was proposed based on an adaptive method to enable particle to carry out different types of search (local or global) at different evolutionary stages. In ALPSO, except the $gbest$ particle, each particle has four learning sources, which produced by the following four learning operators:

Operator a: learning from its $pbest$ position

$$exploitation : v^d_k = \omega v^d_k + \eta \cdot r^d_k \cdot (pbest^d_k - x^d_k) \quad (3)$$

Operator b: learning from a random position nearby

$$jumping\ out : x^d_k = x^d_k + v^d_{avg} \cdot N(0, 1) \quad (4)$$

Operator c: learning from the $pbest$ of its nearest particle

$$exploration: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (pbest_{k\_nearest}^d - x_k^d) \quad (5)$$

Operator d: learning from the $gbest$ particle

$$convergence: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (pbest_{gbest}^d - x_k^d) \quad (6)$$

where $pbest_{k\_nearest}$ is the $pbest$ of the particle closest to particle k, which is better than $pbest_k$; v_{avg}^d is the average velocity of all particles in the d-th dimension, which is calculated by $v_{avg}^d = \sum_{k=1}^{N} |v_k^d|/N$,where N is the population size; $N(0,1)$ is a random number from the normal distribution with mean 0 and variance 1.

The four learning operators play the roles of convergence, exploitation, exploration and jumping out of the basins of attraction of local optima, respectively. In order to enable particles to automatically choose an appropriate learning operator at the appropriate moment during the search process, an adaptive selection mechanism, which is based on the assumption that the most successful operator used in the recent past iterations might be also successful in the future several iterations, is introduced. For each particle, one of the four learning operators is selected according to their selection ratios. The operator that results in a higher relative performance will have its selection ratio increased. Gradually, the most suitable operator will be chosen automatically for a particle and that operator will control the particle's search behavior according to its local fitness landscape at the corresponding evolutionary stage. For all particles, the selection ratio of each operator is equally initialized to 1/4 (except the $gbest$ particle, in which the selection ratios are set to 1/3) and is updated according to its relative performance.

The operators' selection ratios for a particle are updated only if it does not improve for U_f (updating frequency) successive iterations. During the updating period for each particle, the progress value and the reward value of operator i are calculated as follows.

The progress value $p_i^k(t)$ of operator i for particle k at iteration t is defined as:

$$p_i^k(t) = \begin{cases} \mid f(\vec{x}_k(t)) - f(\vec{x}_k(t-1)) \mid, & \text{if operator } i \text{ is chosen} \\ & \text{by } \vec{x}_k(t) \text{ and } \vec{x}_k(t) \text{ is better than } \vec{x}_k(t-1) \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

The reward value $r_i^k(t)$ has three components, which are the normalized progress value, the success rate, and the previous selection ratio. It is defined by:

$$r_i^k(t) = \frac{p_i^k(t)}{\sum_{j=1}^{M} p_j^k(t)}\alpha + \frac{g_i^k}{G_i^k}(1-\alpha) + c_i^k s_i^k(t) \quad (8)$$

where g_i^k is the counter that records the number of successful learning times of particle k, in which its child is fitter than particle k by applying operator i since the last selection ratio update; G_i^k is the total number of iterations where operator i is selected by particle k since the last selection ratio update; $\frac{g_i^k}{G_i^k}$ is the success rate of operator i for particle k; α is a random weight between 0.0 and 1.0; M is the number of operators; c_i^k is a penalty factor for operator i of particle k, which is defined as follows:

$$c_i^k = \begin{cases} 0.9, & \text{if } g_i^k = 0 \text{ and } s_i^k(t) = \max_{j=1}^{M}(s_j^k(t)) \\ 1, & \text{otherwise} \end{cases} \quad (9)$$

and $s_i^k(t)$ is the selection ratio of operator i for particle k at the current iteration. Based on the above definitions, the selection ratio of operator i for particle k in the next iteration $t+1$ is updated according to the following equation:

$$s_i^k(t+1) = \frac{r_i^k(t)}{\sum_{j=1}^{M} r_j^k(t)}(1 - M * \gamma) + \gamma, \quad (10)$$

where γ is the minimum selection ratio for each operator, which is set to 0.01.

For the $gbest$ particle in ALPSO, it will be updated once a particle gets better over time by extracting useful information from that improved particle. Algorithm 1 describes the update framework of the $gbest$ particle.

Algorithm 1 UpdateGbest(particle k,fes)

```
for each dimension d of gbest do
  if rand() < P_l^k then
    x_{t_gbest} := x_{gbest};
    x_{t_gbest}[d] := x_k[d];
    Evaluate x_{t_gbest};
    fes++;
    if f(x_{t_gbest}) < f(x_{gbest}) then
      x_{gbest}[d] := x_{t_gbest}[d];
    end if
  end if
end for
```

where P_l^k is the probability of particle k to learn from the $gbest$ particle

There are two key parameters in ALPSO: the update frequency (U_f) and the learning probability (P_l). The values of U_f and P_l significantly affect the performance of ALPSO. ALPSO introduces some methods to choose the optimal values of the two parameters. For the parameter of update frequency (U_f), each particle is assigned with a different value of U_f instead of using a same value of U_f for all particles. The value of U_f for particle k is defined by the following equation in ALPSO.

$$U_f^k = \max(10 * exp(-(1.6 \cdot k/N)^4), 1) \quad (11)$$

where N is the population size and U_f^k is the update frequency of particle k. For the second parameter of the learning probability (P_l), each particle k is also assigned with a different learning probability, which is initialized by the following equation:

$$P_l^k = \max(1 - exp(-(1.6 \cdot k/N)^4), 0.05) \quad (12)$$

where N is the population size. To adaptively adjust the value of P_l^k, ALPSO needs to calculate the particle's improvement ratio, which is defined by:

$$IMPR_k(t) = \max(\frac{f(\vec{x}_k(t-1)) - f(\vec{x}_k(t))}{f(\vec{x}_k(t-1))}, 0) \quad (13)$$

Algorithm 2 $UpdateP_l(\ )$

Calculate the improvement ratios for all particles using Eq. (13);
Select the particle m that has the largest improvement ratio;
for each particle k **do**
  Calculate the R_k^m using Eq. (14);
  Generate a uniformly distributed random number $p \in [0, 1]$;
  if $p \leq R_k^m$ **then**
    P_l^k=P_l^m;
  else
    P_l^k=$\max(1 - exp(-(1.6 \cdot k/N)^4), 0.05)$;
  end if
end for

Algorithm 3 Update(operator i, particle k, fes)

if $i = a$ **then**
  Update the velocity and position of particle k using operator a and Eq. (2);
else if $i = b$ **then**
  Update the position of particle k using operator b;
else if $i = c$ **then**
  Choose a random particle j;
  if $f(\vec{x}_{pbest_j}) < f(\vec{x}_{pbest_k})$ **then**
    Update the velocity and position of particle k using operator c' and Eq. (2);
  else
    Update the velocity and position of particle j using operator c' and Eq. (2);
    $k := j$;
  end if
else
  Update the velocity and position of particle k using operator d' and Eq. (2);
end if
fes++;

where $IMPR_k$ is the improvement ratio of particle k between iteration $t-1$ and iteration t.

In order to update the P_l of a particle k in ALPSO, particle m that has the largest improvement ratio in the swarm will be chosen at each iteration and then calculate a learning ratio to the particle m by:

$$R_k^m = IMPR_m/(IMPR_m + IMPR_k) \qquad (14)$$

If a randomly generated number $p \in [0, 1]$ is less than R_k^m, particle k will use P_l^m to update the $gbest$ in Algorithm 1; Otherwise, particle k will use its initial learning probability. The learning probability updating method can be seen in Algorithm 2.

It was report that ALPSO [6] can well balance the behavior of exploitation and exploration for an independent particle in its local search space, also it significantly enhances the performance of PSO in terms of convergence speed and solution accuracy comparing with the other peer algorithms.

III. Adaptive Learning Particle Swarm Optimizer-II

In order to further improve the performance of ALPSO especially on complex multi-modal problems, we introduce several main enhancements in ALPSO-II compared with the ALPSO algorithm. First, two learning operators in ALPSO are replaced by two new learning operators. Second, a monitoring mechanism is introduced to monitor particle's status. Finally, an approach to controlling the number of particles that learn from the global best position (named $abest$ position) is added into ALPSO-II. The aim of all the enhancements is to increase diversity so that ALPSO-II can search far more better solutions in complex fitness landscape. The framework of ALPSO-II is described in Algorithm 4, which is explained in the following sections.

A. Learning Operators in ALPSO-II

In ALPSO-II, we still use four learning operators, but the "learning from the $pbest$ of its nearest particle" operator (exploration operator) is replace with "learning from the $pbest$ of a random particle" operator, and each particle learns from a $gbest$ archive position ($abest$) instead of learning from the $gbest$ particle. The two updated learning operators are defined as follows for particle k:

Operator c': learning from the $pbest$ of a random particle

$$exploration: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (pbest_{rand}^d - x_k^d) \qquad (15)$$

Operator d': learning from the $abest$ position.

$$convergence: v_k^d = \omega v_k^d + \eta \cdot r_k^d \cdot (abest^d - x_k^d) \qquad (16)$$

where the $abest$ position is used to store the best position found by ALPSO-II so far.

In ALPSO-II, the bias learning scheme is also used as in ALPSO where a particle only learns from a $pbest_{rand}$ position that is better than its own historical best position $pbest$. Due to this scheme, more computing resources are given to the badly performing particles to improve the whole swarm. It can be seen in Algorithm 3.

It should be noticed that the $abest$ position in Eq. (16) is different from the $gbest$ particle of the whole swarm. Although it is the same position as the $gbest$ particle in the initial population, it will be updated by Algorithm 1 and get better than the $gbest$ particle. Different from the ALPSO algorithm in [6], all particles in ALPSO-II learn from the $abest$ position including the $gbest$ particle. The position and velocity update is shown in Algorithm 3.

By introducing the new exploration operator, ALPSO-II enables a particle to explore non-searched fitness landscape with a higher probability than the ALPSO [6] since that particle will learn from a random particle instead of its nearest neighborhood. As a result, the new exploration operator will increase diversity and it might improve the global search capability of ALPSO-II.

B. Monitoring Particle's Status

Generally speaking, re-initialization is a common method to increase population diversity in evolutionary algorithms. However, there are some problems while using this method, e.g. how to protect these re-initial individuals from being eliminated since they usually have very bad fitness. When to perform re-initialization is also an open issue. Although the former issue does not happen in PSO since there is no selection operation, we still have the later problem.

There are several ways to check when to perform re-initialization. The first method is to check the population diversity. If the population diversity is less than a threshold, we perform re-initialization. The second is to monitor the $gbest$ particle, if it does not improve for a certain number of iterations, re-initialization can be launched. For PSO algorithms, we can monitor particle's velocity. If the velocity's magnitude of a particle is less than a threshold value, we can re-initialize that particle. Whichever method we use, we have to define a threshold value to perform this operation. However, it is very difficult to get an optimal threshold value for a particular problem. In addition, the threshold values for different problems might be different.

The common problem of the above approaches is that they cannot examine if an individual is in the stage of evolution or in the stage of convergence. If that individual converges, we can perform re-initialization. In order to monitor particle's status, we introduce an approach to check whether a particle is in convergence status.

In ALPSO-II, there is a mechanism of monitoring the performance of the four learning operators. The approach is to monitor the selection ratios of the four learning operators. Once a particle converges on a local optimum and none of the four operators can help it to jump out of that local optimum. Their selection ratios will go back to the initial stage where they have equal values of 1/4. Hence, we can use this information to examine whether a particle is converged or not. By using this approach, we can easily avoid the above problems to re-initialize particles.

To achieve this goal, beside to calculate the normal selection ratios as in ALPSO, we need to create a monitoring selection ratio for each learning operator. In ALPSO-II, every definition and operation to calculate and update the monitor ing selection ratios is the same as in ALPSO to calculate and update the normal selection ratios except calculating the progress value $p_i'^k(t)$ of operator i for particle k at iteration t, which is defined as:

$$p_i'^k(t) = \begin{cases} \mid f(\vec{x}_k(t)) - f(\vec{x}_k^{pbest}) \mid, & \text{if operator } i \text{ is} \\ & \text{chosen by } \vec{x}_k \text{ and } \vec{x}_k \text{ is better than } \vec{x}_k^{pbest} \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

To distinguish the definitions related to updating monitoring selection ratios in the two different algorithms, we put a prime symbol after each definition defined in ALPSO, e.g. $p_i'^k(t)$ and $p_i^k(t)$ represent the monitoring progress value and common progress value of operator i for particle k at iteration t, respectively.

In ALPSO-II, the common selection ratios and the monitoring selection ratios are updated at the same time and once they are updated, all the component parameters are reset to the initial states: progress values, reward values, success rates are set to 0. The re-initialization of a particle is performed once the variance of its monitoring selection ratios is less than a constant value of 0.05. The procedures of the monitoring selection ratios update and re-initialization can be seen in Algorithm 4.

Algorithm 4 The ALPSO-II Algorithm

Generate the initial swarm and set up the initial parameters for each particle;
Set fes :=0, iteration counter t=0;
while $fes < T\_Fes$ **do**
  for each particle k **do**
    Select one learning operator i using the roulette wheel selection mechanism with its selection ratio;
    Update(i, k, fes);
    G_i^k++; $G_i'^k$++;
    if $f(\vec{x}_k(t)) < f(\vec{x}_k(t-1)$ **then**
      g_i^k++, and set $m_k := 0$;
      p_i^k+:=$f(\vec{x}_k(t-1)) - f(\vec{x}_k(t))$;
      Perform $UpdateGbest(k, fes)$ for the $abest$ position;
    else
      $m_k := m_k + 1$;
    end if
    if $f(\vec{x}_k(t)) < f(\vec{x}_{pbest_k})$ **then**
      $g_i'^k$++;
      $p_i'^k$+:=$f(\vec{x}_{pbest_k}) - f(\vec{x}_k)$;
      $\vec{x}_{pbest_k} := \vec{x}_k$;
      if $f(\vec{x}_k) < f(\vec{x}_{abest})$ **then**
        $\vec{x}_{abest} := \vec{x}_k$;
      end if
    end if
    if $m_k \geq U_f^k$ **then**
      Update the common and monitoring selection ratios according to Eq. (10);
      for Each operator j **do**
        p_j^k:=0;g_j^k:=0; G_j^k:=0; $p_j'^k$:=0;$g_j'^k$:=0; $G_j'^k$:=0;
      end for
    end if
    if Var($\vec{s'}_k$) <= 0.05 **then**
      Re-initialize particle k;
    end if
  end for
  $UpdateP_l$();
  t++;
end while
where T_Fes is the total fitness evaluations for a run and Var($\vec{s'}_k$) is the variance of the four monitoring selection ratios for particle k.

C. Controlling the Number of Particles that Learn from the $abest$ Position

In ALPSO algorithm, although performing local search for a particle depends on the performance of the local search operators (e.g. the exploitation operator and the exploration operator), it still has a chance to perform global search. As we know, particles, which are far away from the $abest$ position, may not get benefit by learning from it especially for multi-modal problems. In ALPSO-II, to further balance global search and local search, we only allow a certain number of particles (Q), which are close to the the $abest$ position, to learn from it. That is, ALPSO-II only allows Q particles to use the four learning operators and the other particles do not use the convergence operator. We can see how different values of Q affect the performance of ALPSO-II in the following experimental section.

TABLE I

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

Name	Test Function	n	S	f_{min}
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
Rastrigin	$f_2(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	[-5.12, 5.12]	0
Schwefel	$f_3(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500, 500]	0
Ackley	$f_4(\vec{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	[-32, 32]	0
Rosenbrock	$f_5(\vec{x}) = \sum_{i=1}^{n} 100(x_{i+1}^2 - x_i)^2 + (x_i - 1)^2)$	30	[-2.048, 2.048]	0
Schwefel_2_22	$f_6(\vec{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10, 10]	0
Schwefel_1_2	$f_7(\vec{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	[-100, 100]	0
Schwefel_2_21	$f_8(\vec{x}) = \max_{i=1}^{n} \lvert x_i \rvert$	30	[-100, 100]	0
Penalized_1	$f_9(\vec{x}) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4), y_i = 1 + (x_i + 1)/4$	30	[-50, 50]	0
H_Com	$f_{10}(\vec{x})$ =Hybrid Composition function (F15) in [10]	30	[-5, 5]	0
RH_Com	$f_{11}(\vec{x})$ =Rotated Hybrid Composition function (F16) in [10]	30	[-5, 5]	0

IV. EXPERIMENTAL STUDY

A. Test Functions

To test the performance of ALPSO-II, we chose 27 test functions including the traditional functions, the shifted functions and the rotated shifted functions, which are widely used in the literature [7], [3], [12] as well as the complex hybrid composition functions proposed recently in [2], [10]. The details of test functions f_1 to f_{11} are given in Table I. Functions f_{12}-f_{27} are noisy problems, shifted problems, rotated problems, and rotated shifted problems, which are extended from four selected problems in Table I by (a) adding noises; (b) shifting the landscape; (c) rotating the landscape; and (d) combining shifting and rotating of the landscape. The corresponding functions can be seen in Table II. In Table II, "O" represents the original problems, and "N","S", "R" and "RS" represent the modified problems by adding noisy, shifting, rotating, and combination of shifting and rotating, respectively.

TABLE II

TEST FUNCTIONS OF f_{13} TO f_{27}

	O	N	S	R	RS		O	N	S	R	RS
Sphere	f_1	f_{16}	f_{15}	f_{20}	f_{24}	Schwefel	f_3	f_{17}	f_{12}	f_{22}	f_{25}
Rastrigin	f_2	f_{19}	f_{14}	f_{21}	f_{27}	Ackley	f_4	f_{18}	f_{13}	f_{23}	f_{26}

B. Parameter Settings for the Involved PSO Algorithms

Experiments were conducted to compare ALPSO-II with six PSO algorithms on the 27 test problems in 30 dimensions. The peer algorithms include the cooperative PSO (CPSO-H$_k$) [11], the fully informed PSO (FIPS) [7], the comprehensive learning PSO (CLSPO) [3], the adaptive PSO (APSO) [12], the standard PSO (SPSO) proposed in [9], and the ALPSO algorithm [6]. The configuration of each peer algorithm, which is exactly the same as it appeared in the original paper. For ALPSO-II, η_1, η_2, and ω were set to the same values as used in SPSO. V_{max} was set to the half of the search domain for each test function, which can be seen from Table I. We used different parameter values of Q to test the performance of ALPSO-II.

To fairly compare ALPSO-II with the other six algorithms, all algorithms were run independently 30 times on the 27 test problems in 30 dimensions. The initial population and stop criteria are the same for all algorithms for each run. The maximal number of fitness evaluations (T_Fes) used as the stop criteria was set to 100000 for all algorithms on each test function. The population size was set to 10 for all problems.

C. Results and Discussion

We first test how the parameter Q affects the performance of ALPSO-II. The different values of Q in set 0,1,3,5,7,10 were used. The results is shown in Table III. From the results, we can see that different values of Q give quite different performance on most problems. For functions f_3, f_{21}, and f_{22}, no particle learning from the $abest$ position gives the best results. The Q value of 1 helps ALPSO-II to achieve the best performance on functions f_{12},f_{16},f_{17},f_{18},f_{19},f_{23},and f_{25}. For functions f_{14} and f_{27}, the best performance is obtained by setting the neighborhood size of 3 for the $abest$ position. For functions f_{10} and f_{11}, the optimal value of Q is 5. While 7 is the optimal value of Q for function f_{26}. And the other functions achieve the best performance by setting Q value of 10, which is the population size.

Table IV describes the comparison of the mean and variance of the performance of ALPSO-II and the other six peer algorithms over 30 runs on each test function. A two-tailed T-test with 58 degrees of freedom at a 0.05 level of significance was conducted between ALPSO-II and the other six algorithms and the results are shown in Table V. The performance difference is significant between two algorithms if the absolute value of the T-test result is greater than 2.0. In Table V, the suffix "+","$\sim$" or "-" is attached to the end of each result, which represents whether the performance of ALPSO is significantly better than, statistically equivalent to or significantly worse than the performance of its rival respectively.

From Table IV and Table V, we can see that ALPSO-II has a great improvement compared with ALPSO and also

TABLE III

THE MEAN AND VARIANCE FOR ALPSO-II WITH DIFFERENT VALUES OF Q

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
0	3.19e-04	0.0866856	40.0767	0.0382586	26.3309	0.00641491	6064.91	2.25853	0.00113198
	±0.00492394	±0.872817	±234.696	±0.365471	±45.8418	±0.10128	±13031	±6.99178	±0.00827409
1	1.82e-25	2.63e-05	71.1226	8.77e-14	10.5521	2.33e-15	774.272	0.0940275	2.78e-22
	±4.33e-24	±7.73e-04	±461.993	±1.22e-12	±53.6101	±4.51e-14	±6554.44	±1.36323	±8.19e-21
3	1.05e-37	3.14e-05	206.929	3.34e-14	14.075	2.00e-23	12.5449	0.020876	1.85e-32
	±3.10e-36	±9.15e-04	±974.673	±6.11e-14	±110.567	±5.65e-22	±170.868	±0.127035	±6.14e-32
5	4.35e-44	0.0331653	327.68	3.42e-14	4.72059	3.66e-32	0.265745	0.158123	1.57e-32
	±1.28e-42	±0.978236	±1055.14	±4.60e-14	±31.3837	±1.07e-30	±4.7862	±1.31022	±4.50e-47
7	1.66e-68	3.00e-04	414.535	3.57e-14	5.28914	4.54e-38	0.0476268	0.0711354	1.59e-32
	±3.65e-67	±0.0088403	±1337.36	±5.12e-14	±27.559	±1.34e-36	±0.872162	±0.599369	±5.08e-33
10	5.31e-76	5.92e-17	513.413	2.74e-14	4.53951	9.29e-46	0.0254195	0.00865984	1.59e-32
	±1.57e-74	±1.75e-15	±1233.32	±5.31e-14	±19.1285	±2.65e-44	±0.336377	±0.161593	±5.08e-33
f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
0	164.566	291.695	74.8229	0.0364695	0.659069	0.0045937	0.0946551	104.134	0.753227
	±864.696	±2030.23	±678.928	±0.345891	±2.86712	±0.0993323	±0.222948	±626.772	±1.74489
1	90	124.051	63.1713	1.55e-13	0.0663306	3.19e-25	0.0482378	71.1746	0.115955
	±668.581	±1557.17	±434.616	±2.78e-12	±1.35937	±5.22e-24	±0.0657736	±430.397	±0.15188
3	40.0024	86.6691	194.295	2.90e-14	0.00269966	0	0.0511641	146.356	0.167982
	±268.317	±966.775	±920.474	±4.51e-14	±0.0575712	±0	±0.0934132	±619.848	±0.709299
5	33.3333	86.6667	261.594	2.93e-14	0.266642	0	0.0565925	264.782	0.258714
	±258.199	±966.782	±756.297	±4.31e-14	±2.78712	±0	±0.0994114	±848.96	±1.33818
7	36.6667	134.312	369.496	3.24e-14	0.166786	0	0.0601956	466.056	0.551791
	±435.507	±1667.37	±1040.58	±4.20e-14	±2.0288	±0	±0.10302	±931.616	±3.2679
10	40	106.923	436.871	2.67e-14	0.198992	0	0.0765519	414.723	0.486582
	±363.318	±1085.64	±885.548	±4.73e-14	±2.59454	±0	±0.107511	±943.544	±2.88669
f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
0	0.847822	0.0106357	136.259	5213.58	4.28073	0.0651118	5971.65	20.8468	109.569
	±1.77004	±0.11919	±271.161	±7829.31	±6.84644	±0.906418	±7590.21	±0.521797	±128.283
1	0.0936267	3.56e-23	158.297	5585.91	3.371	1.36e-22	5595.76	20.8049	116.906
	±0.593525	±9.92e-22	±261.794	±7215.21	±5.76803	±2.86e-21	±7438.25	±0.77854	±184.589
3	0.136432	6.78e-35	151.134	6263.7	4.79908	6.01e-15	6345.16	20.7147	105.898
	±1.05414	±1.74e-33	±290.152	±5098.66	±10.2086	±1.77e-13	±5255.19	±0.884154	±148.957
5	0.214971	5.03e-54	152.012	6049.45	5.3534	7.41e-30	6329.44	20.7125	121.169
	±1.72119	±1.41e-52	±254.374	±4999.13	±11.9327	±1.56e-28	±4272.98	±0.814065	±198.013
7	0.24976	1.36e-60	156.398	6060.51	4.13264	4.56e-29	6140.41	20.6834	125.032
	±1.44984	±4.00e-59	±276.979	±7980.64	±6.42352	±9.66e-28	±6874.17	±0.793542	±170.94
10	0.184728	8.52e-71	174.759	5595.8	3.92293	0	6076.07	20.745	107.643
	±1.34848	±2.29e-69	±266.957	±7897.55	±8.06215	±0	±7207.31	±0.592678	±193.516

TABLE V

T-TEST RESULTS OF COMPARING ALPSO-II WITH THE OTHER SIX ALGORITHMS

f	ALPSO	APSO	CLPSO	CPSOH	FIPS	SPSO
f_1	-1.06075$^\sim$	-1.6214$^\sim$	-4.69789$^+$	-2.41383$^+$	-23.9359$^+$	-1.29748$^\sim$
f_2	-1.97963$^\sim$	-6.66404$^+$	-5.27699$^+$	-13.801$^+$	-65.3468$^+$	-25.4596$^+$
f_3	-9.34763$^+$	-1.21912$^\sim$	-5.50877$^+$	-26.9873$^+$	-31.6033$^+$	-51.0314$^+$
f_4	-1.51581$^\sim$	-2.40284$^+$	-5.11173$^+$	-2.05477$^+$	-50.3275$^+$	-17.785$^+$
f_5	-5.98233$^+$	-4.03117$^+$	-25.1538$^+$	-8.09945$^+$	-31.8857$^+$	-7.86615$^+$
f_6	-1.00041$^\sim$	-1.06707$^\sim$	-4.6305$^+$	-3.6539$^+$	-13.8511$^+$	-1.15167$^\sim$
f_7	-4.20748$^+$	2.22762$^-$	-13.9907$^+$	-1.50177$^\sim$	-25.7935$^+$	2.22621$^-$
f_8	1.57979$^\sim$	-6.74729$^+$	-14.8828$^+$	1.5379$^\sim$	-83.2277$^+$	-4.14957$^+$
f_9	1$^\sim$	-1.02882$^\sim$	-3.57904$^+$	-1$^\sim$	-4.62343$^+$	-3.57485$^+$
f_{10}	-2.38093$^+$	-1.62823$^\sim$	-0.0142319$^\sim$	-5.155$^+$	-29.5374$^+$	-35.4169$^+$
f_{11}	-2.14081$^+$	-4.822$^+$	0.019596$^\sim$	-4.47075$^+$	-23.1037$^+$	-27.2468$^+$
f_{12}	-7.31876$^+$	-1.33746$^\sim$	-4.56862$^+$	-24.0973$^+$	-27.2351$^+$	-62.1658$^+$
f_{13}	0.18239$^\sim$	-6.22259$^+$	-1.0046$^\sim$	-1.76531$^\sim$	-22.5143$^+$	-19.3547$^+$
f_{14}	-3.69469$^+$	-8.75107$^+$	-6.59355$^+$	-13.9833$^+$	-46.9044$^+$	-20.335$^+$
f_{15}	0$^\sim$	-2.19825$^+$	-2.52609$^+$	-1.46632$^\sim$	-7.47617$^+$	-9.54448$^+$
f_{16}	-5.6533$^+$	0.810908$^\sim$	11.8988$^-$	14.347$^-$	-27.7775$^+$	-1.25726$^\sim$
f_{17}	-8.16028$^+$	4.43758$^-$	-3.26512$^+$	-24.8334$^+$	-36.6763$^+$	-56.3514$^+$
f_{18}	-8.1065$^+$	-4.82325$^+$	7.74591$^-$	13.4864$^-$	-26.3917$^+$	-16.8855$^+$
f_{19}	-3.35534$^+$	-12.3139$^+$	-5.61884$^+$	-12.3992$^+$	-62.3134$^+$	-21.0353$^+$
f_{20}	-1.00001$^\sim$	-1.27409$^\sim$	-4.47775$^+$	-2.21081$^+$	-9.93093$^+$	-1.00001$^\sim$
f_{21}	-2.68428$^+$	1.06262$^\sim$	2.2155$^-$	-4.8912$^+$	-12.4983$^+$	3.14306$^-$
f_{22}	-6.07757$^+$	-9.48845$^+$	-7.0023$^+$	-11.1986$^+$	-4.73968$^+$	-11.0022$^+$
f_{23}	-4.10924$^+$	-2.8342$^+$	6.89448$^-$	-6.66128$^+$	-24.8334$^+$	-9.96075$^+$
f_{24}	0$^\sim$	-1$^\sim$	-4.58573$^+$	-1.70125$^\sim$	-10.1283$^+$	-4.73096$^+$
f_{25}	-5.99211$^+$	-7.90853$^+$	-6.55136$^+$	-11.5386$^+$	-1.54965$^\sim$	-10.0283$^+$
f_{26}	5.42885$^-$	-11.8899$^+$	-12.3133$^+$	-6.4556$^+$	-10.6357$^+$	-9.90871$^+$
f_{27}	-3.45041$^+$	-6.15082$^+$	-1.72509$^\sim$	-12.5193$^+$	-21.5318$^+$	-9.45723$^+$
STA	16+,10∼,1-	15+,10∼,2-	19+,3∼,5-	19+,6∼,2-	26+,1∼,0-	21+,4∼,2-

outperforms the other algorithms on most test functions. As can be seen from Table V, the performance of ALPSO-II is significantly better than the performance of the other peer algorithms on 15 out of 27 functions.

Fig. 1 shows the comparison of convergence speed of the seven algorithm on 15 selected problems. From the comparison result, ALPSO-II shows fast convergence speed (e.g. on functions f_1, f_3,f_6, and f_{11}) and capability of getting out of local optima (e.g. on functions f_2, f_5,f_{10}, and f_{14}).

V. CONCLUSIONS AND FUTURE WORK

This paper presents an updated version of the ALPSO algorithm based on four leanring operators. In order to increase diversity of ALPSO, we introduce several new approaches in ALPSO-II, including two updated learning operators, re-initialization mechanism, and controlling the neighborhood size of the $abest$ position. From the experimental results, we can draw two conclusions from the comparison of the seven algorithms in terms of convergence speed and solution accuracy. First, ALPSO-II greatly improves the performance of ALPSO. The other one is that ALPSO-II outperforms the other state-of-the-art algorithms on most test problems.

Although ALPSO-II has a good performance compared with the other PSO algorithms, several issues should be addressed in the future. How to adaptively tune the neighborhood size of the $abest$ position is the first issue. Second,

TABLE IV

THE MEAN AND VARIANCE OF THE SEVEN ALGORITHMS

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
ALPSOII	5.31e-76	5.92e-17	40.0767	2.74e-14	4.53951	9.29e-46	0.0254195	0.00865984	1.59e-32
	±1.57e-74	±1.75e-15	±234.696	±5.31e-14	±19.1285	±2.65e-44	±0.336377	±0.161593	±5.08e-33
ALPSO	1.34e-70	0.165827	489.545	3.06e-14	24.1441	3.28e-36	3.01274	4.97e-06	1.57e-32
	±3.73e-69	±2.47076	±1398.71	±3.25e-14	±94.7484	±9.67e-35	±20.9394	±7.19e-05	±4.50e-47
APSO	7.65e-67	1.76802	157.788	0.198783	16.7165	2.85e-35	1.51e-05	0.109862	6.73e-24
	±1.39e-65	±7.82543	±2838.24	±2.44014	±87.0207	±7.87e-34	±2.47e-04	±0.411838	±1.93e-22
CLPSO	6.58e-28	0.862298	157.26	4.64e-14	24.3217	8.07e-18	342.719	8.35236	3.44e-28
	±4.13e-27	±4.81982	±581.893	±9.56e-14	±13.1225	±5.14e-17	±722.481	±16.5354	±2.84e-27
CPSOH	4.43e-28	16.9397	2796.49	3.59e-14	22.0575	1.78e-16	347.567	2.34e-04	0.00345563
	±5.41e-27	±36.2039	±3003.46	±1.08e-13	±60.8597	±1.44e-15	±6825.95	±0.00172627	±0.101927
FIPS	2431.85	192.656	4320.67	10.8636	211.692	17.8312	22488.7	31.7602	41800.1
	±2996.72	±86.9591	±3988.23	±6.36685	±190.669	±37.9714	±25716.5	±11.2515	±2.67e+05
SPSO	1.12e-47	61.7205	5793.69	7.19459	11.924	0.001128	3.12e-05	0.0623961	1.00945
	±2.56e-46	±71.5052	±3317.25	±11.9319	±20.0204	±0.0288894	±7.84e-04	±0.346101	±8.3289
f	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
ALPSOII	33.3333	86.6667	63.1713	2.67e-14	0.00269966	0	0.0482378	71.1746	0.115955
	±258.199	±966.782	±434.616	±4.73e-14	±0.0575712	±0	±0.0657736	±430.397	±0.15188
ALPSO	106.667	311.081	461.118	2.64e-14	1.12762	0	0.100977	533.071	1.22046
	±871.015	±2936.92	±1543.78	±3.26e-14	±8.98039	±0	±0.267188	±1613.12	±4.0159
APSO	80	879.673	382.984	0.755438	1.85726	3.86e-21	0.042652	4.09857	0.708133
	±804.984	±4753.44	±7039.6	±3.58086	±6.25058	±5.18e-20	±0.192236	±116.334	±3.61818
CLPSO	33.5097	85.8032	193.45	2.51e-09	2.91855	7.19e-27	0.0197708	158.015	0.0705996
	±258.73	±868.593	±720.108	±7.38e-08	±13.0437	±8.40e-26	±0.0255626	±655.871	±0.0822234
CPSOH	270.283	743.255	2463.02	0.162784	34.9236	2.34628	0.0147016	2669.17	0.0446544
	±1330.96	±4222.58	±2905.15	±2.71988	±73.6607	±47.1965	±0.0206748	±3055.6	±0.0353483
FIPS	899.526	1097.82	3837.65	12.3237	155.269	6262.96	2339.31	4427.77	12.295
	±825.531	±855.428	±4064.61	±16.1451	±97.6382	±24709.3	±2483.96	±3477.12	±13.6106
SPSO	1015.9	1637.58	5944.94	15.1416	129.68	5013.49	57.1522	5881.63	7.30832
	±776.499	±1372.63	±2756.67	±23.0752	±188.097	±15493.4	±1339.68	±3010.73	±12.5628
f	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
ALPSOII	0.0936267	8.52e-71	136.259	5213.58	3.371	0	5595.76	20.6834	105.898
	±0.593525	±2.29e-69	±271.161	±7829.31	±5.76803	±0	±7438.25	±0.793542	±148.957
ALPSO	0.623781	1.49e-57	169.841	7106.47	5.69469	0	7524.69	20.4355	140.432
	±4.62247	±4.39e-56	±250.277	±4805.87	±15.6501	±0	±5901.55	±1.08866	±254.88
APSO	4.01811	5.05e-49	124.336	8277.67	5.02724	1.06e-11	8294.13	21.1134	174.313
	±9.38165	±1.17e-47	±189.744	±5424.76	±16.2428	±3.13e-10	±6778.97	±0.712975	±292.312
CLPSO	1.05972	2.92e-10	113.986	7249.84	1.78883	4.46e-10	7395.84	21.0282	115.922
	±5.03658	±1.92e-09	±119.995	±3503.26	±3.54214	±2.87e-09	±3217.71	±0.229321	±84.7718
CPSOH	22.5102	1.94e-25	199.695	8961.86	13.9178	0.00219601	9057.52	20.8902	248.576
	±53.322	±2.59e-24	±269.837	±6014	±46.3428	±0.0380737	±4793.71	±0.51237	±301.347
FIPS	186.546	6678.58	259.5	6880.84	14.8485	20813.9	6217.46	20.992	242.08
	±88.2546	±19836	±105.185	±6808.54	±12.3519	±60614.1	±9203.25	±0.320154	±112.309
SPSO	58.3943	3.10e-14	103.376	9005.49	9.89541	8413.14	8614.45	20.9681	198.885
	±81.7474	±9.15e-13	±147.309	±6484.17	±18.4389	±52452.8	±4848.07	±0.29758	±248.837

comparing ALPSO-II with other non-PSO algorithms is also an important work. Finally, exploring new learning operators is also an interesting work.

REFERENCES

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

[2] J.J. Liang, PN. Suganthan, and K. Deb."Novel composition test functions for numerical global optimization," in *Proc. 2005 Symp. Swarm Intelligence*, 2005, pp. 68–75.

[3] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 281–295, June, 2006.

[4] J. Kennedy and R. C. Eberhart. "Particle Swarm Optimization," in *Proc. 1995 IEEE Int. Conf. on Neural Networks*, 1995, pp. 1942–1948.

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

[6] C. Li and S. Yang. "An adaptive learning particle swarm optimizer for function optimization". *Proc. of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 381–388.

[7] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simple, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 204-210, June, 2004.

[8] R. Poli, J. Kennedy, and T. Blackwell,"Particle swarm optimization: An overview," *Swarm Intelligence* , Vol. 1, pp. 33–58, 2007.

[9] Shi, Y. and R. C. Eberhart,"A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 69–73.

[10] 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.

[11] F. van den Bergh, A. P. Engelbrecht,"A Cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 225–239, June, 2004.

[12] Z. Zhan,J. Zhang, Y. Li and H. S. Chung, "Adaptive Particle Swarm Optimization ", *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, pp. 1362–1381, Dec, 2009.

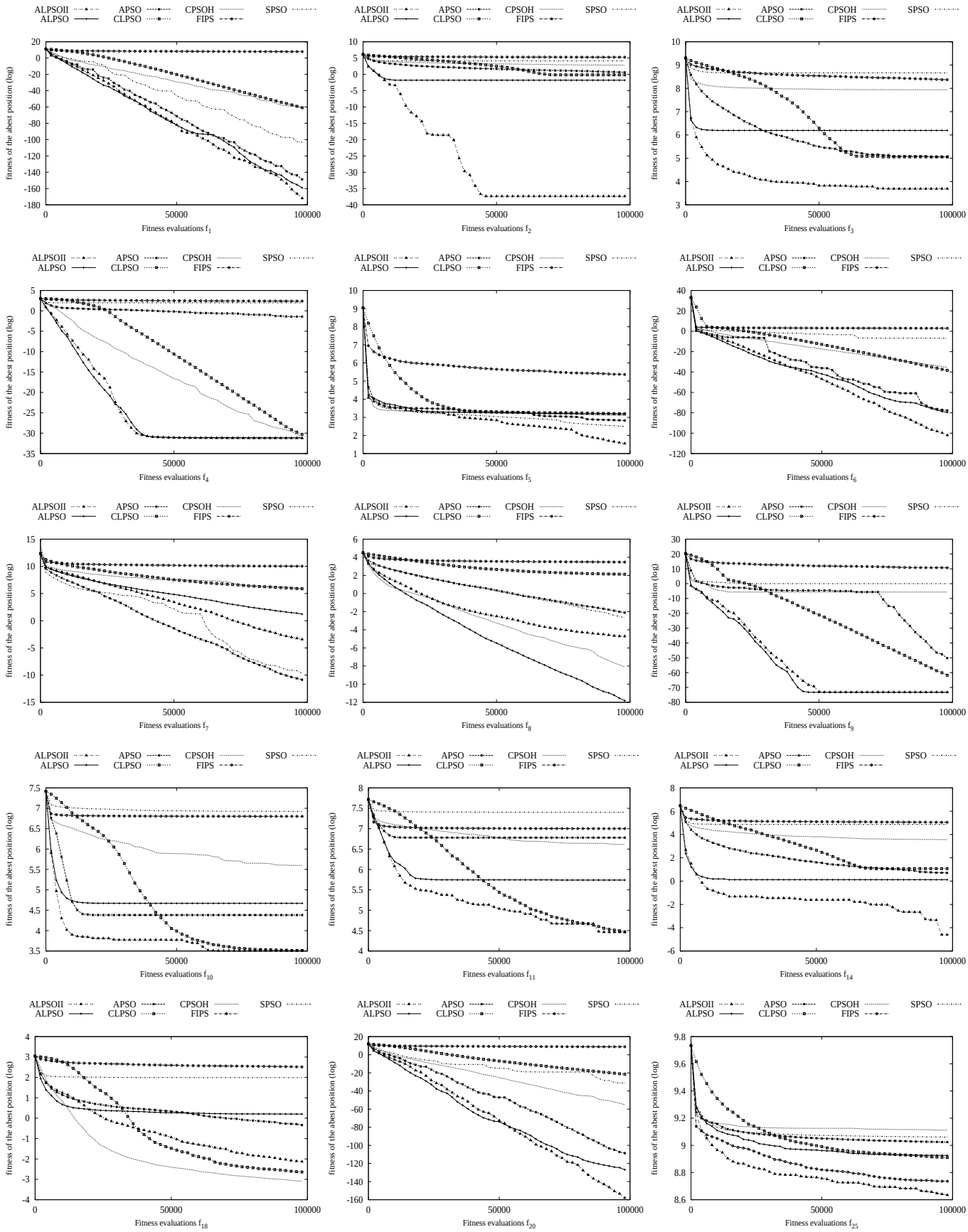

Fig. 1. Evolutionary process of the algorithms on 15 selected functions.