New Learning Automata based Particle Swarm Optimization Algorithms

M. Hamidi[2,3] M. R. Meybodi [1]

[1]Computer Engineering and Information Technology Department, Amirkabir University of Technology Tehran, Iran
[2] Computer Engineering and Information Technology Department, Azad Islamic University Qazvin, Iran
[3] Electronic and Computer Engineering Department, Azad Islamic University, Zarghan, Iran
(mandana.hamidi@gmail.com, mmeybodi@aut.ac.ir)

Abstract: Particle swarm optimization (PSO) is a population based statistical optimization technique which is inspired by social behavior of bird flocking or fish schooling. The main weakness of PSO especially in multimodal problems is trapping in local minima. Recently a learning automata based PSO called PSO-LA to improve the performance of PSO has been reported. PSO-LA uses one learning automaton for configuring the behavior of particles and also creating a balance between the process of global and local search. Although PSO-LA produces better results than the standard PSO but like standard PSO it may trap into local minima. In this paper four improvements on PSO-LA are proposed. These improvements are proposed to reduce the probability of trapping PSO-LA into local minima. Unlike PSO-LA which uses one learning automaton to guide all particles, in the proposed PSO algorithms one learning automaton is assigned to each particle as the particle brain which controls the particle movement in the search space. The proposed algorithms are tested on 8 benchmark functions. The results have shown that the proposed PSO algorithms are superior to standard PSO, PSO with inertia weight (PSOw) and previously reported LA based PSO algorithms.

Keywords: Particle Swarm Optimization, Learning Automata, PSO-LA, Function Optimization

1. Introduction

Particle Swarm Optimization (PSO) method is a heuristic search/optimization technique and was first proposed by Kennedy and Eberhart[1] in 1995. PSO is motivated from the collective behavior of social animals, phenomena such as bird flocking, fish schooling etc. as well elements of social psychology. One of the drawbacks of standard PSO model is premature convergence and trapping in local optima

Many researchers have worked on improving PSO model performance in various ways and developed many interesting variants such as *single-solution PSO* algorithms in unconstrained, static ,continuous optimization problems[1][2][3], *niching PSO*, *constrained PSO*, *multi-Objective PSO*, *dynamic-environment PSO* and *discrete PSO*[4][5][6]. Single-solution PSO algorithms can be divided into the following categories: PSO algorithms in which one or more parameters inertia weight, cognitive and social weights are adapted. [7]-[11]. Social-based algorithms including variant social topology among particles **Error! Reference source not found.**[16]. Sub-swarm-based algorithms that contains more than one swarm, cooperating [17]or even competing with each other[18][24]. Repelling methods which avoid particle collision, or repel one particle from another to improve diversity[20][15]. Memetic algorithms which combine local search mechanisms with the basic PSO[19][21]. hybrid algorithms, which combine concepts from other methods such as evolutionary computing, genetic algorithms[22], evolutionary programming, fuzzy systems [23] ant colony, learning automata and cellular learning automata [28]. Learning automaton (LA) is a general-purpose

stochastic optimization tool, which has been developed as a model for learning systems. The LA tries to determine, iteratively, the optimal action to apply to the environment from a finite number of actions that are available to it. The environment returns a reinforcement signal that shows the relative quality of an action of the LA, then LA adjusts itself by means of a learning algorithm. LA have been used to enhance the learning capability of other algorithms such as neural networks [25][26], genetic algorithms [35] and particle swarm optimization[5][29]. Recently four solutions based on LA for solving two drawbacks of standard PSO model: premature convergence and trapping in local optima [5][27][29] have been reported. In [5], a discrete version of PSO based on LA is proposed, which the set of LA is assigned to a particle may be viewed as the brain of the particle determining its position from its own and other particles past experience. In [35], the LA based PSO reported in [5] is combined with the neighborhood topology concept and a new discrete PSO method based on Cellular Learning Automata (CLA) is introduced. In the proposed algorithm each cell of CLA contains a particle. In [34], combination of CLA and continuous PSO algorithm is proposed. In this model, each cell of CLA contains a population of particles (sub-swarm) and a learning automaton. LA of each cell is responsible for configuring the behavior of the cell's particles. In [27], an LA based PSO method called PSO_LA is reported. In PSO_LA an LA takes is responsible to configure the behavior of particles in order to create a balance between the process of global and local searches(For more information about PSO-LA the reader may refer to .section 3).Although PSO-LA algorithm produces better results than the standard PSO [28] but like standard PSO it may trap into local minima. Another problem with PSO-LA is that particles must do whatever LA has decided. Sometimes a particle has a good behavior on the search space, but because LA has chosen action "*Follow the best*" it must follow the group, that is it will be forced to change its search behavior and as a result losing its useful search information.

In this paper four improvements on PSO-LA[27] called LAPSO1, LAPSO2, LAPSO3 and LAPSO4 are proposed. These improvements are proposed to reduce the probability of trapping PSO-LA into local minima. In LAPSO1 unlike PSO-LA which updates particles velocity and particles oscillation domain when LA selects "*Continue your way*" action, it updates particles velocity and particle's oscillation domain when it selects any one of its actions. In LAPSO2, LAPSO3 and LAPSO4 unlike LAPSO1 which uses one learning automaton to guide all particles, one LA is assigned to each particle, as a brain of particle, for regulating its movement during the search process. Using one LA for each particle has two advantages: particles do not lose their useful search information and particle's variation will be increased which increases the probability of escaping from local minima be increased. For preventing from trapping in the local minima, in LAPSO3 and LAPSO4 ,when then the particle's velocity becomes less than a threshold, it will be reset randomly. In order to evaluate the performance of the proposed algorithms they are tested on 8 benchmark functions. The results have shown that the proposed PSO algorithms are superior to standard PSO, PSOw and previously reported LA based PSO algorithm (PSO-LA). One of the most advantages of the proposed algorithms, which could be observed from the result, is that they are invariant to swarm size, can search space and escape from local minima even with the minimum swarm size.

The rest of the paper is organized as follows. PSO is reviewed briefly in Section 2. In section 3, the subject of learning automata and PSO-LA are described,. The proposed improvements on PSO-LA are given in section 4. Section 5 is devoted to simulation results and finally section 6 concludes the paper.

2. Particle Swarm Optimization

Particle swarm optimizer (PSO) simulates the behaviors of the birds flocking. In PSO, each solution is a point in the search space and may be regarded as a bird. We refer to the bird as a particle in the algorithm[1]. All particles have fitness values and velocities. The particles fly through the D dimensional problem space by learning from the best experiences of all the particles. Therefore, the particles have a tendency to fly towards

better search areas over the course of search process. The velocity V_i^d and position X_i^d updates of the dth dimension of the ith particle are given in equations (1) and (2). [1][2]

$V_i^d(t+1) = WV_i^d(t) + c_1 rand1_i^d(t)(pbest_i^d(t) - X_i^d(t)) + c_2 rand2_i^d(t)(gbest^d(t) - X_i^d(t))$	(1)
$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1)$	(2)

where c_1 and c_2 are the acceleration constants representing the weighting of stochastic acceleration terms that pull each particle towards pbest and gbest positions. $rand1_i^d$ and $rand2_i^d$ are two random numbers in the range [0,1]. $X_i = (X_i^1, X_i^2, ..., X_i^D)$ is the position of the ith particle , $V_i = (V_i^1, V_i^2, ..., V_i^D)$ is the velocity of the ith particle, $pbest_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ is the best previous position yielding the best fitness value pbesti for ith particle and $gbest = (gbest^1, gbest^2, ..., gbest^D)$ is the best position discovered by the whole particles. W is the inertia weight used to balance the global and local search abilities.

3. Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments. An LA has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment. The aim is to learn to choose the optimal action, the action with the highest probability of being rewarded, through repeated interaction of the system. If the LA is chosen properly, the process of interacting with the environment can result in selection of the optimal action. A study reported in [30] illustrates how a stochastic automaton works in feedback connection with a random environment.

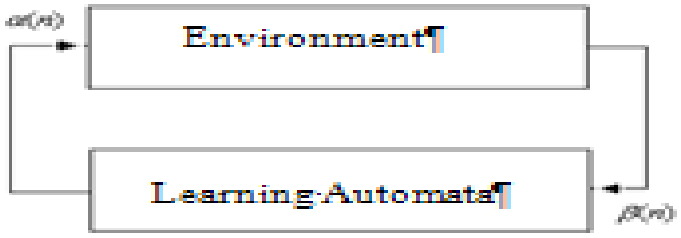


Figure 1: The interaction between learning automata and environment

A VSLA is a quintuple <*α,β,p,T(α,β,p)*>, where *α, β, p* are an action sets with *s* actions, an environment response set and the probability set *p* containing *s* probabilities, each being the probability of performing every action in the current internal automaton state, respectively. *T* is the reinforcement algorithm, which modifies the action probability vector *p* with respect to the performed action and received response. Let a VSLA operate in an environment with *β={0,1}*. A general linear schema for updating action probabilities can be represented as follows. Let action *i* be performed.

If $\beta(t)=0$,

$$p_i(t+1) = p_i(t) + a[1 - p_i(t)] \quad , \quad p_{j\neq i}(t+1) = (1-a)\,p_j(t)$$

If $\beta(t)=1$,

$$p_i(t+1) = (1-b)\,p_i(t) \quad , \quad p_{j\neq i}(t+1) = (b/s-1) + (1-b)\,p_j(t) \qquad (3)$$

Where *a* and *b* are reward and penalty parameters. When *a=b*, automaton is called L_{RP}. If *b=0* and *0<b<<a<1*, the automaton is called L_{RI} and $L_{R\varepsilon P}$, respectively. The overall operation of Learning Automaton

is summarized in the algorithm of Figure 2. For more information about learning automata the reader may refer to .[26][25]

Initialize p to [1/s,1/s,…,1/s] where s is the number of actions
While not done
  Select an action *i* based on the probability vector **p**
  Evaluate the action and return a reinforcement signal β
  Update the probability vector using the learning algorithm
End While

Figure 2: Psudocode of variable-structure learning automaton

4. PSO-LA

PSO-LA uses a learning automaton (LA) to configure the behavior of the particles in PSO in order to create a balance between the process of global and local search[27]. The LA has two actions: "*Follow the best*" and "*Continue your way*". If LA selects the "*Follow the best*" action, all particles in the swarm must follow the best visited position by itself , i.e. its own experience (the best self experience and best team experience) and the position of the best particle (local Search) both determine the velocity of the particle in the next iteration. If LA selects "*Continue your way*" no particle changes its velocity and continues its own way (global search). PSO-LA starts with initializing the position and velocity of the particles. Then until the desired goal is met or a maximum number of iterations is reached the following steps are repeated.

1- LA selects one of its actions based on its probability vector.
2- Based on the selected action, the method of velocity updating for particles is selected and then the particles update their velocities and positions.
3- According to particles updating results a reinforcement signal is generated which will be used to update the probability vectors of the learning automaton.

The selected action of LA in any iteration specifies the velocity updating method of particles for that iteration. Selection of "Follow the best" action means that only the best self experience and best team experience determine the velocity of the particles for the next iteration (equation (4)). The current particle's velocity is ignored. If the "Continue your way" action is selected, the new velocity of the particle will be equal to its current velocity.

$$V_i^d(t+1) = c_1 rand_1(t)(pbest_i(t) - X_i^d(t)) + c_2 rand_2(t)(gbest_i(t) - X_i^d(t)) \quad (4)$$

Evaluating the selected action is done by comparing each particle's new position with its old position. If a specific percent of population (C_{imp}) are improved, the selected action will be evaluated as appropriate and otherwise evaluated as inappropriate action. C_{imp} is a parameter of the PSO-LA that must be properly set.

Although PSO-LA algorithm produces better results than the standard PSO [33] but like standard PSO it may trap into local optima. Another problem with PSO-LA is that particles must do whatever LA has decided. Sometimes a particle has a good behavior on the search space, but because LA has chosen action "*Follow the best*" it must follow the group, that is it will be forced to change its search behavior and as a result losing its useful search information.

5. The Proposed LA based PSO Algorithms

In this paper four improvements on PSO_LA [27] called LAPSO1, LAPSO2, LAPSO3 and LAPSO4 are proposed. These improvements are proposed in order to solve some of the problems of PSO-LA mentioned previously.

5.1 LAPSO1: In PSO-LA1 similar to PSO-LA, a learning Automaton which has two actions "*Follow the best*" and "*Continue your way*" is used to configure the particles search behavior. In PSO-LA unlike PSO-LA which updates particles velocity and particles oscillation domain when LA selects "*Continue your way*" action, it updates particles velocity and particle's oscillation domain when it selects any one of the actions "*Continue your way*" and "Follow the best". Algorithm LAPSO1 is given below.

1- Initialize the position and velocity of the particles. Initialize LA's probability vector
2- Repeat until the desired goal is met or maximum number of iterations is reached.
   a. LA selects one of its actions based on its probability vectors.
   b. If LA selects "*continue your way*", then the velocity will be updated as equation (4)
   c. If LA selects action "*Follow the best*" then the velocity of the particles will be updated using equation(5)

$$w(t) = w_0 - \frac{t * w_1}{MaxGen}$$
$$V_i^d(t+1) = w(t) * V_i^d(t) + c_1 * rand_{1_i}^d * (pbest_i^d(t) - X_i^d(t)) + c_2 * rand_{2_i}^d * (gbest^d(t) - X_i^d(t)) \quad (5)$$

   d. Update positions of the particles using equation
3- Generate reinforcement signal β as described below
4- Update the probability vector of the learning automaton.

Evaluating the selected action taken by learning automata and then generating the reinforcement signal is done by comparing each particle's new position with its old position. If a specific percent of population (C_{imp}) are improved, the selected action will be evaluated as appropriate and otherwise evaluated as inappropriate action. As in PSO-LA, here C_{imp} is a parameter of the algorithm that must be properly set.

5.2 LAPSO2: As mentioned before, one of the PSO-LA weakness is immolation of useful particle search behavior. LAPSO2 is proposed to solve this weakness. In LAPSO2 one learning automata is assigned to each particle. Each learning automata again has two actions "*Follow the best*" and "*Continue your way*". Algorithm LAPSO2 is given below.

1. Initialize positions and velocities of the particles. Initialize LA's probability vector.
2. Repeat until the desired goal is met or maximum number of iterations is reached.
3. All particles perform steps "a" through "h" in parallel
   a. The learning automaton associated to the particle chooses one of its actions
   b. If selected action is "*continue your way*" then the velocity will be updated according to equation (4)
   c. If selected action is "*Follow the best*" then the velocity will be updated according to equation(5)
   d. Update the particle's positions according to equation (2)
   e. Generate reinforcement signal β using equation (6) given below

$$\beta = \begin{cases} 0 & if \quad fitness \quad (X_i(t+1)) > fitness \quad (X_i(t)) \\ 1 & otherwise \end{cases} \quad (6)$$

   f. Update the learning automaton probability vector

5.3 LAPSO3: In PSO-LA, LAPSO1 and LAPSO2, when the cognitive component and social component of the velocity equation are too small and particles follow the best particle position in the group,

then, particles velocity will decrease. Also when the cognitive and social component of velocity equation are near to zero and knowing the fact that the value of inertia weight is smaller than 1 then the particles velocity decrease rapidly and causes the particles movement to become very slow. LAPSO3 is proposed as a remedy to this problem. LAPSO3 algorithm is similar to LAPSO2 except that when the particles velocity reaches a threshold Vc a new velocity is computed using equation(7) and assigned to the particle. This way by reinitializing the velocity of lazy particles (particles with small velocity) trapping into local minima that may happen in Modified-PSO-LA1 will be prevented.

$$V_{i+1}^{d} = \begin{cases} V_{i+1}^{d} & if \ |V_{i+1}^{d}| \geq Vc \\ u(-1,1)V\max/\rho & if \ |V_{i+1}^{d}| < Vc \end{cases} \quad (7)$$

5.3 LAPSO4: LAPSO4 is similar to LAPSO2 except the way that the reinforcement signal is computed. In LAPSO4 reinforcement signal β is generated according to equation (8) given below.

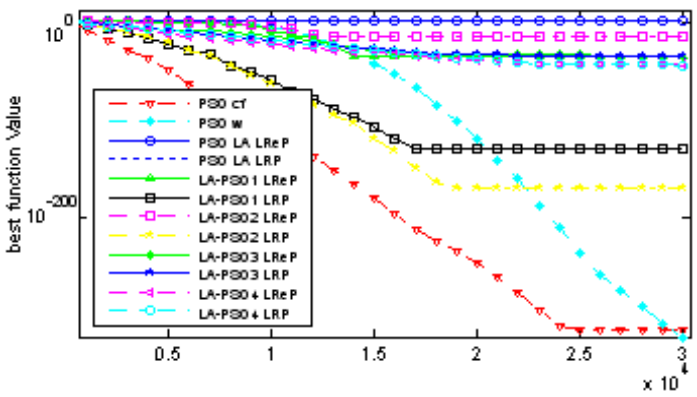
$$\beta = \begin{cases} 0 & if \quad fitness \quad (X_i(t+1)) > pbest_i \\ 1 & otherwise \end{cases} \quad (8)$$

6. Simulation Results

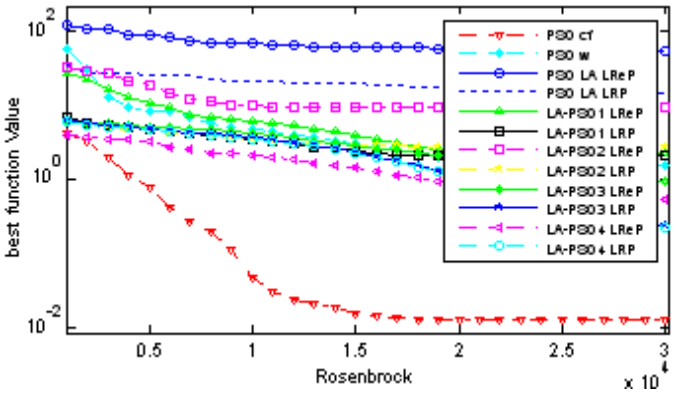
In order to show the performance of the proposed algorithms, they are used for optimization of eight benchmark functions. These functions contain two unimodal functions (Sphere, Rosenbrock) and six multimodal functions (Ackley, Griewanks, Weierstrass, Rastrigin, Noncontinuous Rastrigin and Schwfel). The equation and the shape of these functions in 2 dimensional search spaces are given in Table 1.

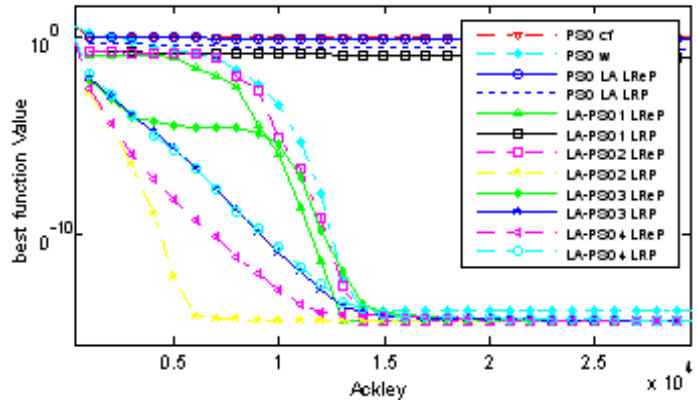
The global optima x^*, the corresponding fitness value $f(x^*)$, the search ranges $[X_{min}, X_{max}]$, and the initialization range of each function are given in Table 2. The experiment were conducted to compare proposed algorithms with standard PSO and PSOw, on ten and 30 dimensional versions of eight test functions. The values of parameters for different algorithms are given in Table 3. All the proposed algorithms are tested with both L_{RP} and $L_{R\varepsilon P}$ learning algorithms. Parameters for these two learning algorithms are given in the last two columns of Table 3. The naming convention used for the proposed algorithms is as follow: for example LA-PSO2-L_{ReP} is LA-PSO2 in which L_{ReP} has been used as the learning algorithm for updating the action probability vector. For the experimentation, a maximum fitness evaluation (FEs) is set to 30'000. As we wish to show our methods are swarm-size invariant, so the population size is set to 5.

Table 4 gives the results for 10 dimensional versions of test functions. The means and the variances of 30 runs for twelve algorithms and eight functions are reported and best results are shown in bold.

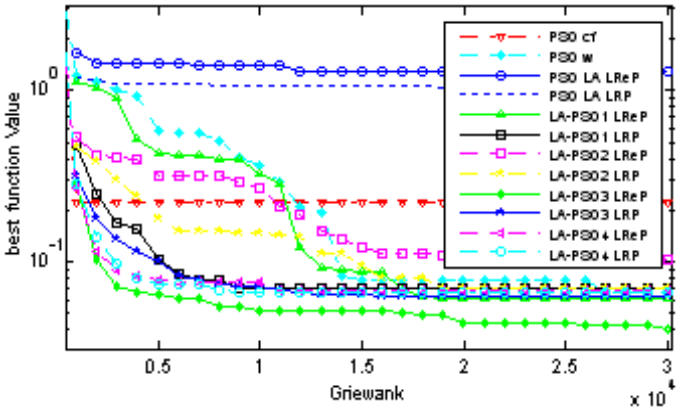

best function Value
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP

(a)

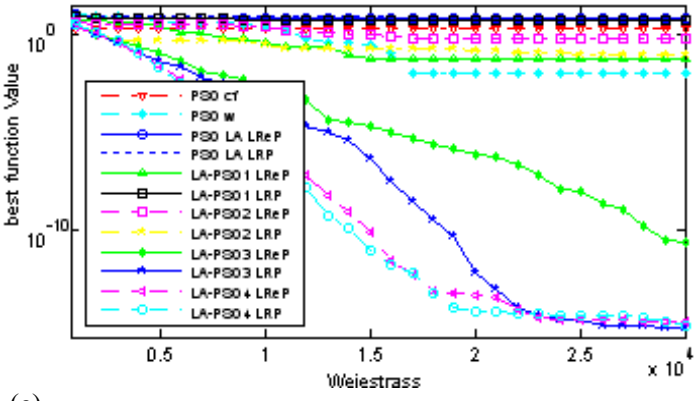

best function Value
Rosenbrock
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP


best function Value
Ackley
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP

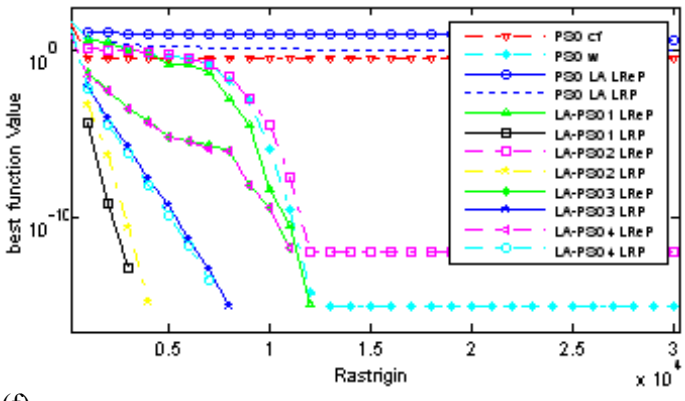
(c)


best function Value
Griewank
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP

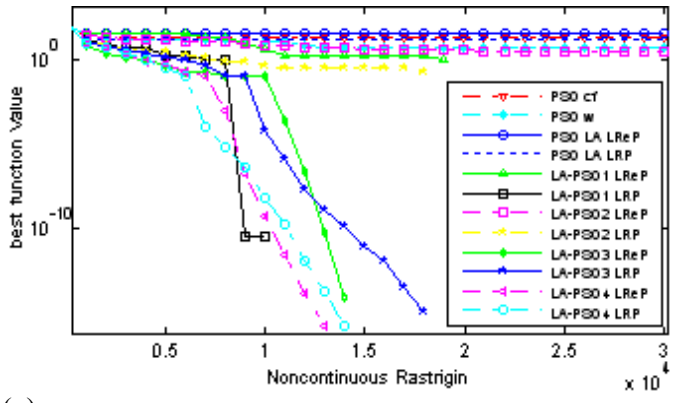
(d)


best function Value
Weiestrass
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP

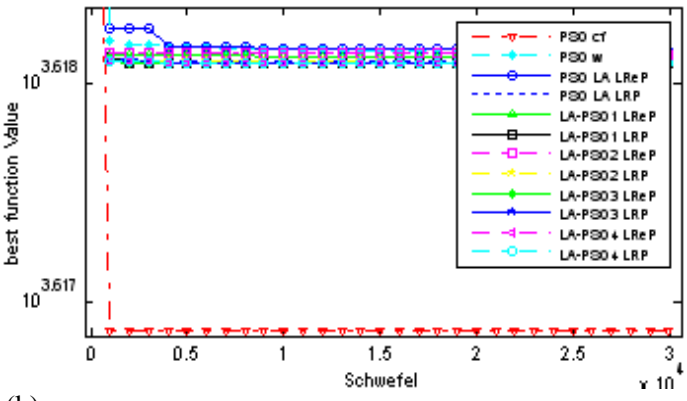
(e)


best function Value
Rastrigin
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LA-PSO1 LReP
LA-PSO1 LRP
LA-PSO2 LReP
LA-PSO2 LRP
LA-PSO3 LReP
LA-PSO3 LRP
LA-PSO4 LReP
LA-PSO4 LRP

(f)



(g)



(h)

Figure 3 presents the convergence characteristics in terms of the best fitness value of the median run of each algorithm for each test function. By looking at the results given in Table 4 for 10-dimensional version, one may conclude that the proposed algorithms produce better results for all test functions except for Sphere function for which PSO with inertia weight (PSOw) performs better.

By looking at the results one may conclude that all the proposed algorithm produce better results for function 2 and five multimodal functions. LAPSO2-L_{RP} produces the best result for function 3. LAPSO3 and LAPSO4 perform better than other algorithms on functions 4, 5, 6 and 7 and none of the twelve algorithms finds acceptable result for function 8. However, the proposed algorithms (LAPSO3 and LAPSO4) find the global minima when other algorithms miss the global optimum basin. From the results we can say that PSOw performs better than the standard PSO and converges faster than other algorithms LAPSO-1 through LAPSO4) on unimodal functions. On multimodal functions PSO-LA, standard PSO and PSOw trap in local minima but LAPSO1, LAPSO2, LAPSO3 and LAPSO4, can escape from local minima and have higher exploration ability (search large area of search space). LAPSO3 and LA-PSO4 has the best result comparing to standard PSO, PSOw, PSO-LA, LAPSO1 and LAPSO2.

The experiments conducted on 10-D problems are repeated on 30-D problems and the results are given in Table 5. From the results we can say that the proposed algorithms (LAPSO1 through LAPSO4) obtain similar ranking as in the 10-D problems. LAPSO3-L_{RP} and LAPSO4-L_{RP} surpass all other algorithms on functions 3 ,4 ,5 ,6 and 7. As a whole the result for a 30_D function is not as good as the results for its 10-D counterpart. LAPSO4-L_{RP} performs the best on functions 3, 4 and 5. The experimentations have shown that using L_{RP} learning algorithm in the proposed algorithms always produces better results. The experimentations have also shown that the proposed algorithms perform better on multimodal function. Another point noted about the proposed algorithms is that they keep higher diversity among particles (even when the number of particles is small) comparing to standard PSO, PSOw, PSO_{cf} and PSOLA.

7. Conclusions

In this paper four new learning automata based PSO algorithms were proposed. These algorithm are improved versions of a previously reported learning automata based PSO, called PSO-LA. These improvements are proposed to reduce the probability of trapping PSO-LA into local minima. Unlike PSO-LA which uses one learning automaton to guide all particles, in the proposed PSO algorithms one learning automaton is assigned to each particle to control the particle movement in the search space. From the results of experimentations on 8 benchmark function optimization problems, we observed that the proposed PSO algorithms, especially LAPSO3 and LAPSO4, are superior to standard PSO, PSO with inertia weight

(PSOw) and previously reported LA based PSO algorithms for multimodal functions and can achieve best result even with small size swarm.

Table 1: Test functions

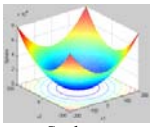
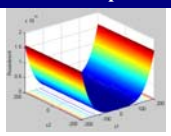
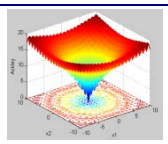
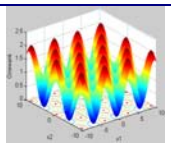
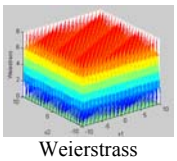
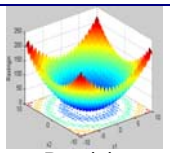
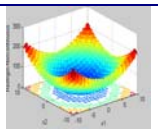
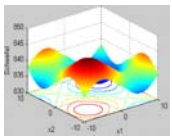
Equation	2 D shape	Equation	2 D shape
$f_1(x) = \sum_{i=1}^{D} x_i^2$	 Sephere	$f_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1}^2)^2 + (x_i^2 - 1)^2)$	 Rosenbrock
$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\Pi x_i)\right) + 20 + e$	 Ackley	$f_4(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	 Greiwank
$f_5(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k(x_i + 0.5))]\right) - D\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k 0.5)], a = 0.5, b = 3, k\max = 20.$	 Weierstrass	$f_6(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	 Rastrigin
$f_7(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10)$ $y_i = \begin{cases} x_i & \lvert x_i\rvert < 1/2 \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert >= 1/2 \end{cases}$ $fori = 1,2,...,D$	 NonContinuous Rastrigin	$f_8(x) = 418.9829 * D - \sum_{i=1}^{D} x_i \sin\left(\lvert x_i\rvert^{\frac{1}{2}}\right)$	 Schwefel

Table 2: global optimum, search ranges and initialization ranges of the test functions

F	X*	F(x*)	Search Range	Initialization Range
F1	[0,0,…,0]	0	$[-100,100]^D$	**$[-100,50]^D$**
F2	[1,1,…,1]	0	$[-2.048, 2.048]^D$	**$[-2.048, 2.048]^D$**
F3	[0,0,…0]	0	$[-32.76, 32.76]^D$	**$[-32.768, 16]^D$**
F4	[0,0,…0]	0	$[-600, 600]^D$	**$[-600, 200]^D$**
F5	[0,0,…0]	0	$[-0.5, 0.5]^D$	**$[-0.5, 0.2]^D$**
F6	[0,0,…0]	0	$[-5.12, 5.12]^D$	**$[-5.12,2]^D$**
F7	[0,0,…0]	0	$[-5.12, 5.12]^D$	**$[-5.12, 2]^D$**
F8	[420.96,… 420.96]	0	$[-500.500]^D$	**$[-500.500]^D$**

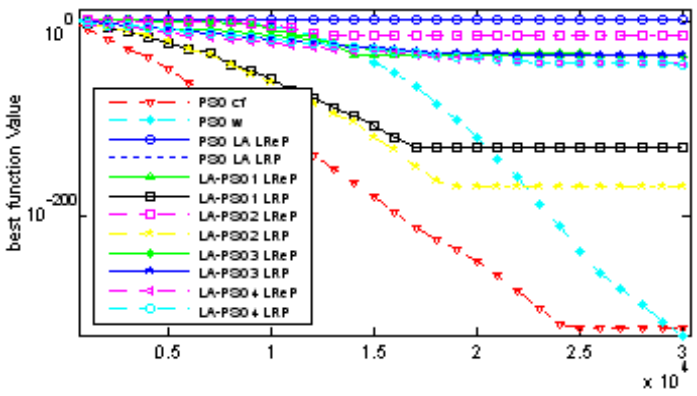
Table 3: parameter setting for the Involved PSO algorithms

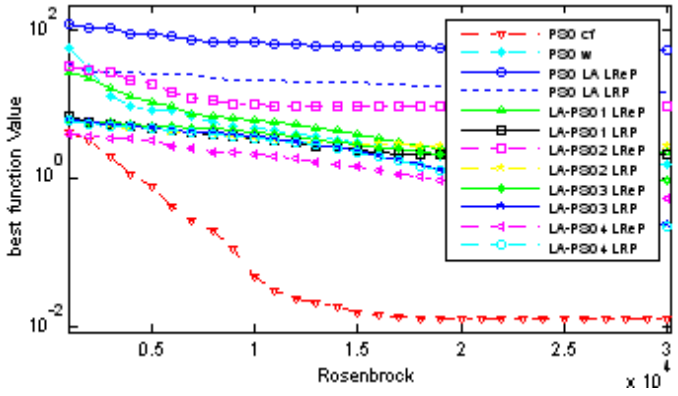
	W0	W1	C1	C2	Cimp	Vc	ρ	LA_LRP	LA_LReP
PSO w	0.9	0.4	1.45	1.45					
PSO_{cf}	٠.75	0.75	2	2					
PSO-LA	0.9	0.4	1.45	1.45	75			α=β=0.01	**α =0.001 , β =0.01**
LAPSO1	0.9	0.4	1.45	1.45	75			α=β=0.01	**α =0.001 , β =0.01**
LAPSO2	0.9	0.4	2	2				α=β=0.01	**α =0.001 , β =0.1**
LAPSO3	0.9	0.4	2	2		1e-20	7	α=β=0.01	**α =0.001 , β =0.01**
LAPSO4	0.9	0.4	2	2		1e-20	7	α=β=0.01	**A=0.001 , β =0.01**

Table 4: Results for 10-D problems

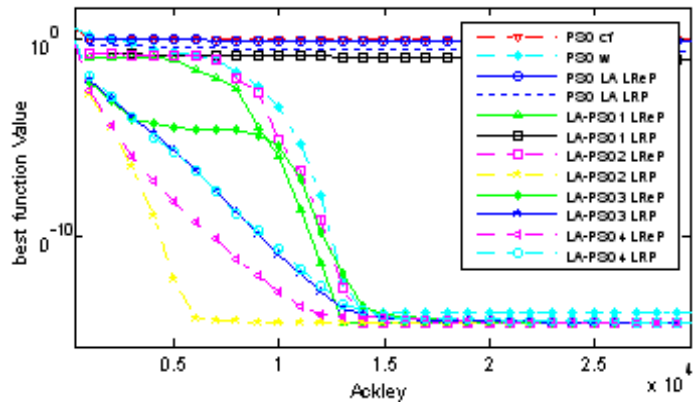
	F1	F2	F3	F4
PSO cf	1.2418e-314±4.5618e-371	**0.0124±3.9667e-004**	2.2737e-014±8.5323e-028	0.0799±0.0016
PSO w	**2.0405e-321±4.3906e-257**	1.5283±2.6112	0.6957±0.6130	0.1632±0.0085

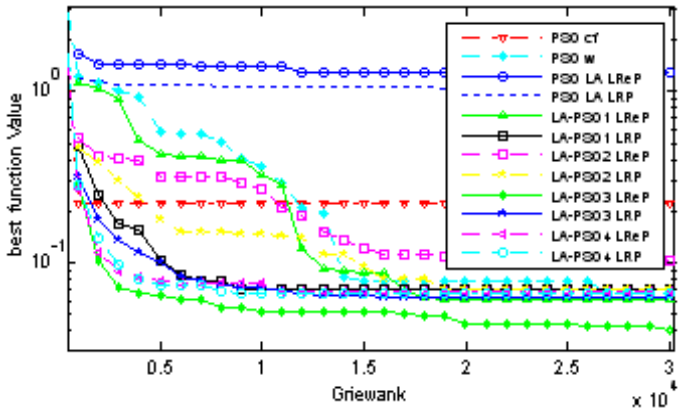
PSO LA LRP	8.3992±7.3897	14.0788±6.4807	0.2463±0.0046	0.9643±0.0362
PSO LA LReP	56.3939±134.7245	52.9057±201.8486	0.7809±0.00978	1.2837±0.109
LAPSO1 LRP	3.0470e-129±9.1730e-257	2.0069±1.9846	0.0977±0.0197	0.0692±3.3223e-004
LAPSO1 LReP	1.6983e-034±2.8842e-067	2.1958±4.8066	3.5527e-015±0	0.0615±0.0012
LAPSO2 LRP	3.4075e-170± 3.4075e-190	2.6870±2.7885	**3.1974e-015±1.2622e-030**	0.0711±0.0016
LAPSO2LReP	1.4721e-014±2.1672e-027	9.0277±35.4323	3.5527e-015±0	0.1018±0.0023
LAPSO3 LRP	1.6675e-034±1.2507e-068	0.2280±0.0078	3.5527e-015±0	0.0625±9.3018e-004
LAPSO3 LReP	1.4987e-034±1.2208e-068	0.9110±0.4825	3.5527e-015±0	**0.0406±5.4235e-004**
LAPSO4 LRP	3.3601e-044±5.2617e-088	0.0208±0.2195	3.9080e-015±1.2622e-030	0.0671±4.4587e-004
LAPSO4 LReP	5.1131e-044±2.0892e-087	0.5263±1.4968	3.5527e-015±0	0.0674±0.0012
	F5	**F6**	**F7**	**F8**
PSO cf	0.0333±0.0068	3.5527e-016±1.6097e-31	0.9000±0.7667	4.1507e+003±0.0030
PSO w	2.4964±1.5691	0.8379±0.9089	21.7368±65.8713	**4.138e+003±374.6959**
PSO LA LRP	7.3139±0.4251	0.9815±0.0333	37.8118±0.01	4.1505e+003±0
PSO LA LReP	6.6444±0.124	4.4543±0	14±2.45	4.1510e+003±0
LAPSO1 LRP	5.2655±13.1760	**0±0**	**0±0**	4.1504e+003±0
LAPSO1 LReP	0.0532±0.0011	**0±0**	**0±0**	4.1506e+003±0
LAPSO2 LRP	0.1032±0.020	**0±0**	**0±0**	4.1504e+003±1.6673e-004
LAPSO2LReP	0.6085±0.55791	7.7645e-013±5.7292e-024	3.2005±6.8477	4.1507e+003±0.0095
LAPSO3 LRP	**1.0658e-015±2.9451e-030**	**0±0**	**0±0**	4.1504e+003±0
LAPSO3 LReP	2.4722e-011±5.7176e-021	**0±0**	**0±0**	4.1504e+003±1.9734e-004
LAPSO4 LRP	1.4211e-015±1.1780e-029	**0±0**	**0±0**	4.1504e+003±6.1485e-005
LAPSO4 LReP	2.1316e-015±3.1414e-029	**0±0**	**0±0**	4.1504e+003±0





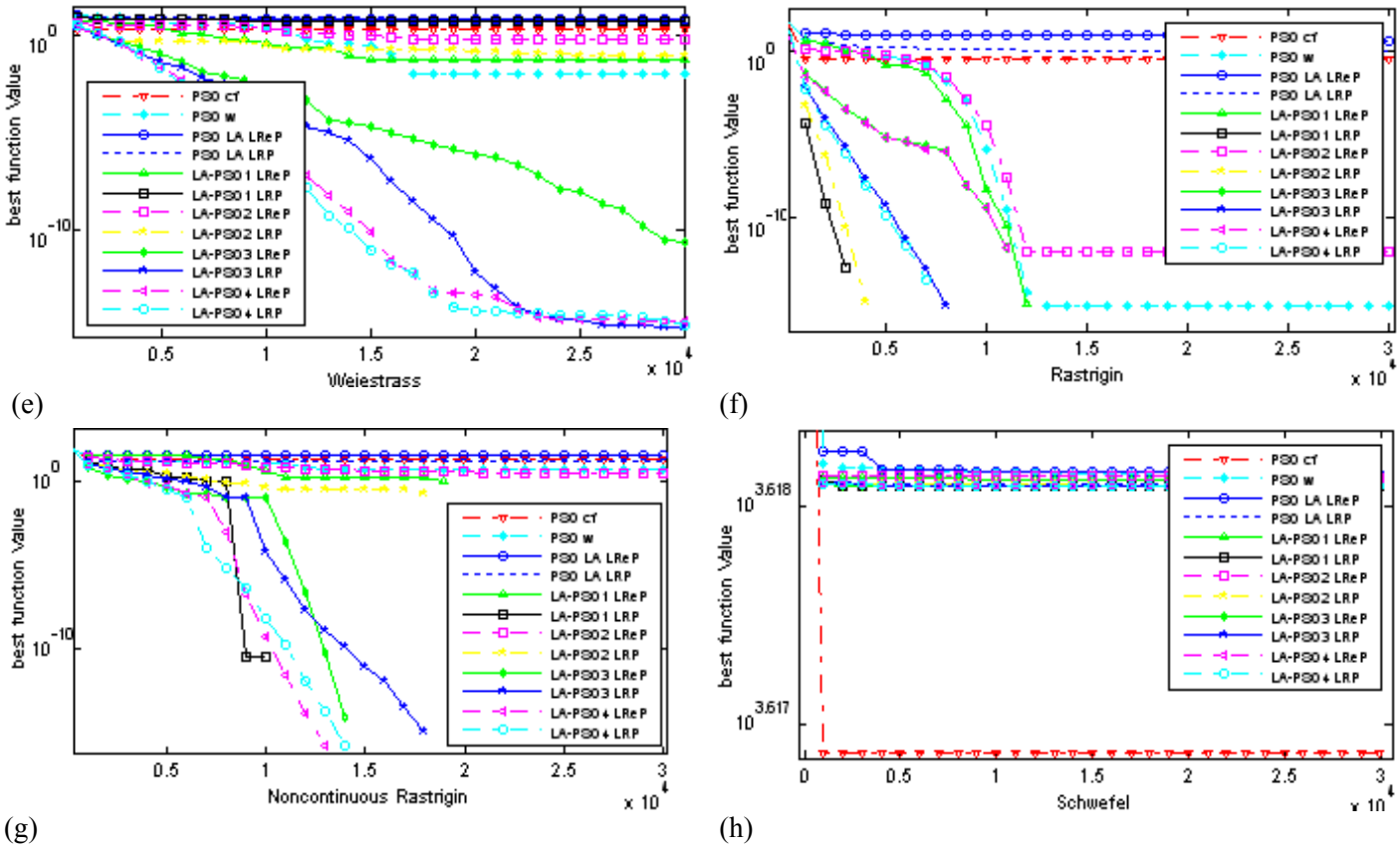
(a)





(d)

(c)



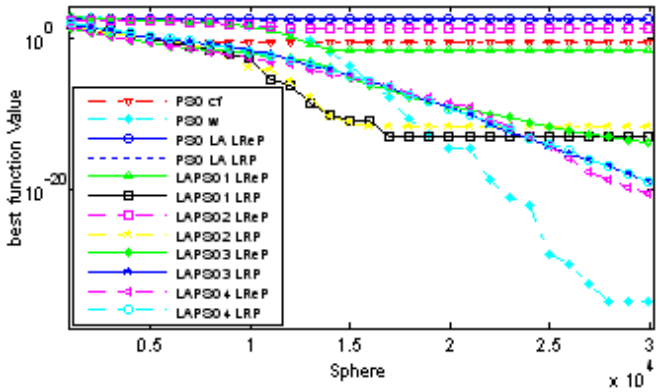
(e)

(f)

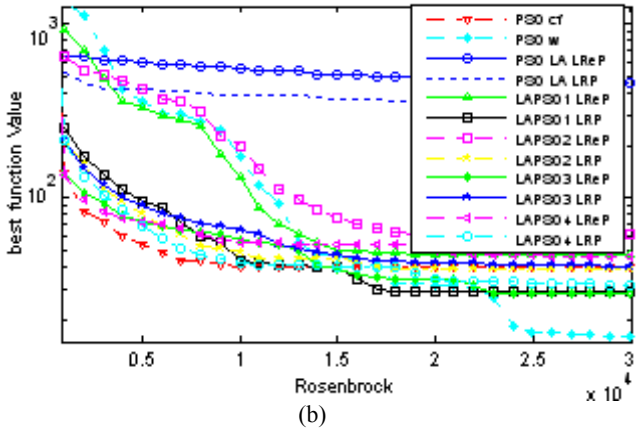
(g)

(h)

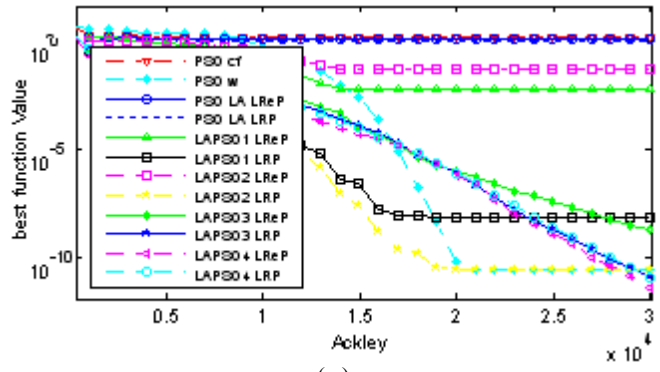
Figure 3The median convergence characteristics of 10-D test functions. (a) Sphere function. (b) Rosenbrock's function. (c) Ackley's function. (d) Griewank's function. (e) Weierstrass function. (f) Rastrigin's function. (g) Noncontinuous Rastrigin's function. (h) Schwefel's function


best function Value
Sphere
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

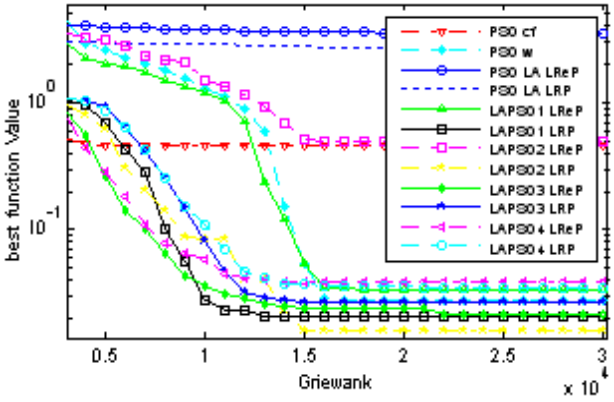
(a)


best function Value
Rosenbrock
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

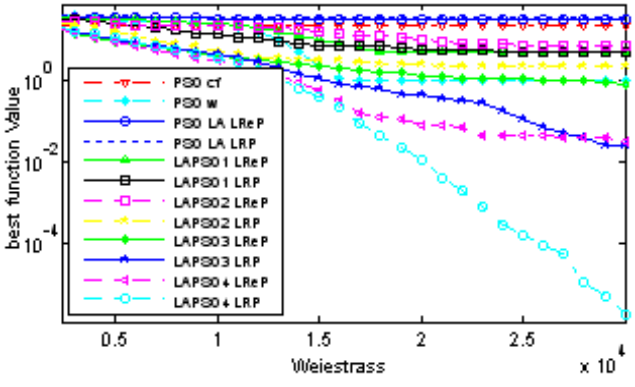
(b)


best function Value
Ackley
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

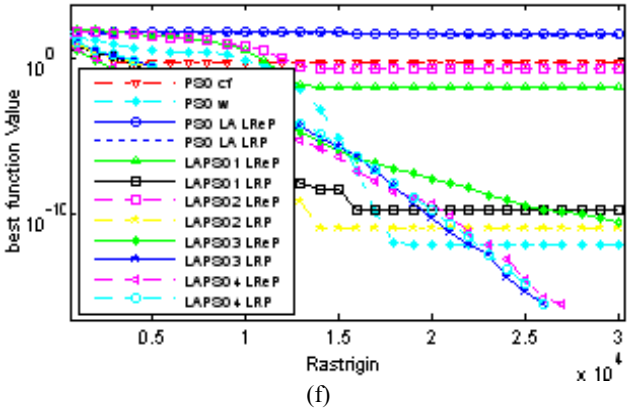
(c)


best function Value
Griewank
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

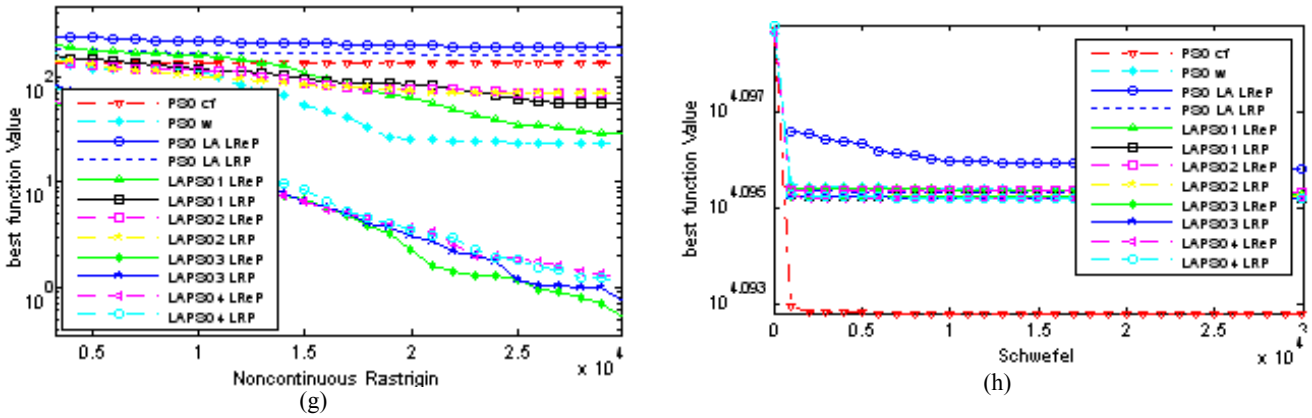
(d)


best function Value
Weiestrass
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

(e)


best function Value
Rastrigin
x 10^4
PSO cf
PSO w
PSO LA LReP
PSO LA LRP
LAPSO1 LReP
LAPSO1 LRP
LAPSO2 LReP
LAPSO2 LRP
LAPSO3 LReP
LAPSO3 LRP
LAPSO4 LReP
LAPSO4 LRP

(f)



(g) (h)

Figure 4The median convergence characteristics of 30-D test functions. (a) Sphere function. (b) Rosenbrock's function. (c) Ackley's function. (d) Griewank's function. (e) Weierstrass function. (f) Rastrigin's function. (g) Noncontinuous Rastrigin's function. (h) Schwefel's function

Table 5: Results for 30-D problems

	F1	F2	F3	F4
PSOcf	0.4298±0.5723	40.0853±644.5545	1.5869±0.4308	0.4444± 0.1245
PSO w	**4.217e-034±3.6705e-067**	**17.7492±79.5980**	2.7320e-008±7.4076e-015	0.0329±0.0018
PSO LA LRP	258.8194±1.2231e+003	335.4866±1.8071e+003	1.0370±0.0052	3.4420±0.0155
PSO LA LReP	383.1832±1.3331e+003	460.5667±4.5067e+003	1.2550±0.0043	2.6987±0.1291
LAPSO1 LRP	1.401e-013±7.1681e-026	29.3942±275.0456	8.1099e-009±2.4196e-016	0.0204±0.0010
LAPSO1 LReP	0.0272± 0.0055	47.0219±666.8714	0.0061±8.7614e-005	0.0334±4.8586e-004
LAPSO2 LRP	2.167e-012±1.3394e-023	39.020±674.8228	2.6823e-011±1.9782e-021	**0.016±6.2225e004**
LAPSO2LReP	18.1432±3.2883e+003	61.7932±1.9677e+003	0.0511±0.0138	0.4893±1.3695
LAPSO3 LRP	8.770e-020± 2.3656e-039	40.3704±902.2664	1.1636e-011± 1.4016e-022	0.0272±7.3246e-004
LAPSO3 LReP	1.430e-014±1.9797e-027	28.0745 ±591.2671	1.9474e-009±1.9474e-009	0.0214±3.24623-004
LAPSO4 LRP	1.347e-019±2.1415e-038	31.5803±877.3246	1.2714e-011±5.4917e-023	0.0346±0.0011
LAPSO4 LReP	2.488e-021±4.6186e-041	46.1007±782.2940	**3.8639e-012± 4.7452e-023**	0.0386±0.0022
	F5	**F6**	**F7**	**F8**
PSO cf	20.7938±15.6794	0.6344±0.2649	132.0500±841.2194	**1.2383e+004±5.4533e+003**
PSO w	1.0371±1.0632	1.1958e-012±4.0718e-024	23.4±291.8222	1.2454e+004±0.1690
PSO LA LRP	37.0573±37.0573	33.3892±14.6191	157.3910±195.6791	1.2454e+004±0.2034
PSO LA LReP	31.5882±21.5825	44.0232±27.6941	194.8198±81.5650	1.2469e+004±17.7907
LAPSO1 LRP	4.9605±10.6045	1.7110e-10±1.3083e-019	54.9038±678.2944	1.2452e+004±0.0219
LAPSO1 LReP	4.7831±31.7103	0.0141±0.0017	28.3307±968.3732	1.2454e+004±0.1595
LAPSO2 LRP	2.3808±4.7863	1.7110e-010±1.3083e-019	68.3213±1.0758e+003	1.2452e+004±0.0079
LAPSO2LReP	7.2227±5.7324	0.2013±0.1300	65.7398±462.8724	1.2454e+004±5.2434
LAPSO3 LRP	0.0244±0.0050	1.1782e-011±1.3750e-021	0.7140±1.1523	1.2451e+004±0.0033
LAPSO3 LReP	0.8174±0.7905	2.9972e-011±7.4394e-021	**0.5034±0.2814**	1.2451e+004±0.4467
LAPSO4 LRP	**1.278e-006±1.1917e-011**	**0±0**	1.0136±1.7770	1.2454e+004± 9.3635e-004
LAPSO4 LReP	0.0314±0.0064	**0±0**	1.1746±2.6184	1.2454e+004±0.0021

References

[1] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948, 1995.

[2] K. E. Parsopoulos, V. P. Plagianakos, G.D. Magoulas, and M. N. Vrahatis, "Objective Function "Stretching" to Alleviate Convergence to Local Minima", NonLinear Analysis, Theory, Methods and Applications, Vol. 47, No. 5 pp. 3419-3424, 2001.

[3] C. Coello Coello and M. Lechuga, "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimizations", In Congress on Evolutionary Computation, Piscatawa, New Jersey, USA, Vol. 2, pp. 1056-1051, IEEE Service Center, 2002.
[4] X. Hu and R. C. Eberhart, “Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization” , in Proc. Congr. Evol. Comput., Honolulu, HI pp. 1677–1681, 2002.
[5] R. Rastegar, M. R. Meybodi and K. Badie, "A New Discrete Binary Particle Swarm Optimization based on Learning Automata", In Proceedings of International Conference on Machine Learning and Applications(ICMLA2004),16-18December 2004, pp.456-462,USA, IEEE Press, 2004.
[6] J. Kennedy and R. C. Eberhart, “A Discrete Binary Version of the Particle Swarm Algorithm”, Proceedings of the International Conference on Systems, Man, and Cybernetics, IEEE Service Center, Piscataway, NJ, pp. 4104-4108, 1997.
[7] J. Peng, Y. Chen, and R. C. Eberhart, "Battery Pack State of charge Estimator Design Using Computational Intelligence Approaches", Proceedings of the annual Battery Conference on Applications and Advances, pp. 173-177, 2000.
[8] P. Yin, " A Discrete Particle Swarm Algorithm for Optimal Polygona Approximation of Digital Curves", Journal of Visual Communication and Image Representation, Vol. 15, No. 2, pp.241-260, June 2004.
[9] A. Ratnaweera, S. Halgamuge, and H. Watson, “Self-organizing hierarchical particle swarm optimizer with time varying accelerating coeffi- cients,” IEEE Trans. Evol. Comput., vol. 8, pp. 240–255, June 2004.
[10] S. Naka, T. Genji,T. Yura, and Fukuyama, "Particle Distribution State Estimation using Hybrid Particle Swarm Optimization", In IEEE power Engineering Society Winter Meeting, Vol. 2, pp. 815-820, January 2001.
[11] G. Venter , R. T. Haftka and J. Sobieszczanski-Sobieski, "Multidisciplinary Optimization of a Transport Artificial Wing using Particle Swarm Optimization", Proceedings of Ninth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002.
[12] Y. Zheng, L. Ma, L. Zhang, and Qian, "On The Convergence Analysis and Parameter Selection in Particle Swarm Optimization", Proceeding of the International Conference on Machine Learning and Cybernetics, Volum 3,pages 1802-1807, November 2003.
[13] A. Ratnaweera, S. Halgamuge, and H. Watson, “Particle Optimizer with Time varying Acceleration Coefficients", Proceedings of the international Conference on Soft Computing and Intelligent Systems, pages 240-25, 2002.
[14] A. Ratnaweera, S. Halgamuge, and H. Watson, “Particle Swarm Optimization with Self-Adaptive Acceleration Coefficients" Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery, pages 2411-2418 December 2003.
[15] D. Braendler and T. Hendtless. Improving Particle Swarm Optimization the Collective Movement of Swarm, IEEE Transactions and Evolutionary Computation , in press
[16] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1931-1938, July 1999.
[17] F. Van den Bergh and A. P. Engelbrecht, "Effects of Swarm Size on Cooperative Particle Swarm Optimizers", Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, USA, pp. 892-899, 2001.
[18] A. Silva, A. Neves, and E. Costa, "An Empirical Comparison of Particle Swarm and Predator Prey Optimization", Proceedings of the Thirteenth Irish Conference on Artificial Intelligence and Cognitive Science , Lecture Notes in Artificial Intelligence , Vol. 2464, pp. 103-110 , Springer, Verlag, 2002.
[19] J. S. Vesterstrom , J. Giget and T. Krink, "Division of Labor in Particle Swarm Optimization", Proceedings of the IEEE Congress on Evolutionary Computation, pages 1570-1575 IEEE press, 2002.
[20] Y. Chunming Yang and Dan Simon, "A New Particle Swarm Optimization”, Technique Electrical and Computer Engineering Department Cleveland State University Cleveland, Ohio, 2004.
[21] P. Yin, " A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves", Journal of Visual Communication and Image Representation, Vol. 15, No. 2, pp. 241-260, June 2004.
[22] M. Løvberg، T. Rasmussen and T. Krink, "Hybrid Particle Swarm Optimizer with Breeding and Subpopulation", Proceedings of the Third Genetic and Evolutionary Computation Conference, Vol. 1, pp. 476-460-476, 2001.
[23] Y. Shi and R. C. Eberhart, “A Modified Particle Swarm Optimizer”, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[24] A. Jensen and S. Kristensen, "Basic PSO versus Multi Swarm PSO", Topics of Evolutionary Computation, EVALife, Department of Computer Science, University of Aarhus, Denmark, 2002.
[25] M. R. Meybodi and H. Beigy, "A Note on Learning Automata Based Schemes for Adaptation of BP Parameters", Journal of Neurocomputing, Vol. 48, No. 4, pp. 957-974, October 2002.
[26] H. Beigy and M. R. Meybodi, "A Learning Automata Based Algorithm for Determination of Minimum Number of Hidden Units for Three Layers Neural Networks", Journal of Amirkabir, Vol. 12, No. 46, pp. 111-136, 2001.
[27] M. Sheybani, and M.R. Meybodi, "PSO-LA: A New Model for Optimization", Proceedings of 12th Annual CSI Computer Conference of Iran, Shahid Beheshti University, Tehran, Iran, pp. 1162-1169, Feb. 20-22, 2007.
[28] M. Sheybani and M. R. Meybodi, "CLAPSO: A New Model for Optimization", Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Volume on Computer, Telecommunication Research Center, Tehran, Iran, May 15-17, 2007.
[29] N. Jafarpour and M.R. Meybodi, "A Hybrid Method for Optimization (Discrete PSO + CLA)", Proceedings of International Conference on Intelligence and Advance Systems(ICIAS2007), Kuala Lumpor, Malaysia, Nov. 25-28, 2007.
[30] K. S. Narendra and M. A. L Thathachar., Learning Automata: An Introduction, Prentice-Hall Inc, 1989
[31] Baluja, S., Caruana, R., "Removing The Genetics from The Standard Genetic Algorithm", Proceedings of ICML'95, PP. 38–46, Morgan Kaufmann Publishers, Palo Alto, CA, 1995.
[32] X. H. Wang and J. J. Li, "Hybrid Particle Swarm Optimization with Simulated Annealing", Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, 2004.
[33] G. Y. Gary and Lu. Haiming, "Dynamical Population Strategy assisted Particle Swarm Optimization", Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, Texas, pp. 697-702, October 2003.
[34] L. Schoofs and B. Naudts, "Swarm Intelligence on the Binary Constraint Satisfaction problem", Proceedings of the IEEE Congress on Evolutionary Computation 2002 Honolulu, Hawaii USA, 2002.
[35] M. Munetomi, Y. Takai, and Y. Sato, "StGA: An Application of Genetic Algorithm to Stochastic Learning Automata", Syst. Comput. Jpn., Vol. 27, pp. 68-78, 1996.