Improving Learning Automata based Particle Swarm: An Optimization Algorithm

Mohammad Hasanzadeh, Mohammad Reza Meybodi and Saeed Shiry Ghidary
Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
mdhassanzd@aut.ac.ir, mmeybodi@aut.ac.ir, shiry@aut.ac.ir

Abstract— Numerous variations of Particle Swarm Optimization (PSO) algorithms have been recently developed, with the best aim of escaping from local minima. One of these recent variations is PSO-LA model which employs a Learning Automata (LA) that controls the velocity of the particle. Another variation of PSO enables particles to dynamically search through global and local space. This paper presents a Dynamic Global and Local Combined Particle Swarm Optimization based on a 3-action Learning Automata (DPSOLA). The embedded learning automaton accumulates the information from individuals, local best and global best particles then combines them to navigate the particle through the problem space. The proposed algorithm has been tested on eight benchmark functions with different dimensions. The work is unique from its test bed; evaluations contain large population size (150) and high dimension (150). The results show that, fitness and convergence pace is better than traditional PSO, DGLCPSO and previous PSO based LA algorithms.

I. INTRODUCTION

One of the common methods for optimization of continuous nonlinear functions is particle swarm optimization (PSO) [1], [2]. This algorithm has two main concepts: artificial life and collective intelligent of brutes. The inertia weight (w) [3–5] is one of the PSO parameters to bring about a balance between the exploration and exploitation characteristics of PSO. Cooperative PSO (CPSO) is a variation on standard PSO which used multiple swarms [6]. Recently it is proposed a combined dynamic global and local particle swarm optimization (DGLCPSO) algorithm to improve the performance of original PSO [7].

Learning automaton (LA) [8] is a multi-propose casual tool, which is an expansion model for learning machines. PSO, like other stochastic search methods, is highly sensitive to adjustment of affective parameters. Recently a LA based PSO model [9–12] called PSO-LA has been reported to improve the performance of PSO. In [10] a PSO-LA model is proposed in which the LA is responsible for configuring the behavior of the swarm and also balancing the process of global and local search. In [11] a Cellular LA (CLA) based discrete PSO is introduced as a solution of premature convergence. To reduce the probability of trapping PSO-LA into local minima, in [12] four modifications on PSO-LA model are proposed. Also the ability of LA for adaptive PSO parameter selection studied in [13], [14].

In an attempt to enhance the performance of DGLCPSO we introduce a dynamic global and local combined particle swarm optimization based on 3-action learning automata (DPSOLA). We use the three dynamic searching Strategies of DGLCPSO as 3 actions of LA to control the path and velocity of the particles.

The formula used in previous PSO-LA models [9–12] for updating the velocities and trajectories of particles is based on standard PSO [1], [2]. Also this work differ from the existing ones by its dynamic global and local combined particle swarm optimization iterative formula, in which all particles dynamically share the best information of themselves, local particle and global particle. To control the behavior of whole swarm, one LA is used in [10]. This kind of using LA; easily increase the probability of trapping particles in local minima or even worse than expected, avoid the particles to continue their suitable search path in order to coordinate with the swarm search path. In [12] one LA is assigned to each particle and tests run on small number of particles and dimensions but large number of iteration. Although a small number of iterations are needed for convergence, the proposed algorithm is tested on eight benchmark functions with large number of particles and dimensions. Experimental results indicate that the DPSOLA algorithm improves the performance on the benchmark functions significantly.

The rest of this paper is organized as follow: The next section introduces the standard PSO and DGLCPSO algorithm. LA and PSO-LA models are presented in section 3. The DPSOLA algorithm and its area of improvements will be noted in section 4. Section 5 is devoted to parameter settings and simulation results. Finally section 6 concludes the paper.

II. DYNAMIC GLOBAL AND LOCAL PSO ALGORITHM

PSO is an optimization algorithm which is based on extrinsic behavior of population. In standard PSO the particles are manipulated by the following equations [2]:

$$v_{id}(k+1) = wv_{id}(k) + c_1 r_1 (p_{id}(k) - x_{id}(k)) + c_2 r_2 (p_{gd}(k) - x_{id}(k)) \quad (1)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

In PSO algorithms, each solution is like a ‘‘fish’’, and each fish ‘‘swim’’ around in the multidimensional problem space with an acceleration. For more information about standard PSO the reader may refer to [2].

978-1-4577-0045-3/11/$26.00 ©2011 IEEE

In the rest of this section we review a Dynamic Global and Local Combined Particle Swarm Optimization (DGLCPSO) algorithm [7]. There are three types of information involved in DGLCPSO: best information of own particle, local particle and global particle. During the search process each particle shares this information by their associated particles.

Assume that we have a D-dimensional search space and *m* particles. The *i*th particle is presented as a D-dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD}); i = 1, 2..., m$. It means that the *i*th particle specifies the position X_i in the search space and the particle's position indicate probable solution. We can calculate the particle's fitness by putting its position in the objective function. Likewise the *i*th particle's velocity is presented as $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Denote the best position of the *i*th particle as $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$ and the best position of particle's local neighborhood as $P_l = (p_{i1}, p_{i2}, ..., p_{iD})$ and the best position of global space as $P_g = (p_{i1}, p_{i2}, ..., p_{iD})$. After calculating these three best values, the velocity and position of each particle is updated according to the following equations [7]:

$$\begin{aligned} v_{id}(k+1) = {} & wv_{id}(k) \\ & + r_1(a + 1/(endgen + 1 - k)) * (p_{id}(k) - x_{id}(k)) \\ & + (b - 1/(endgen + 1 - k)) * (p_{id}(k) - x_{id}(k)) \\ & + cr_2(p_g(k) - x_{id}(k)) \end{aligned} \quad (3)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (4)$$

$$p_l(k+1) = pl(k-1, k-2, ...1) \cup p_l(k) \quad (5)$$

Where $w \in [0,1]$ is inertia weight which controls the degree of global and local search. k is the current generation number. $a, b \in [0.6, 1.2]$ are weights index, and *endgen* represent the maximum number of iterations. The constant c is acceleration constant that controls the movement of particle in a single iteration.

III. Learning Automata: Applications in PSO

A. Learning Automata (LA)

Learning Automaton [8], [13] is a machine that can perform number of finite actions. Each selected action is evaluated by a probabilistic environment and the result of evaluation as a feedback, returns to the automaton in form of a positive or negative signal, and consequently the automaton takes this signal into account, in the next action selection phase. Learning how to select the best action from a finite set of actions is the final target of automaton. The best action is an action that maximizes the probability of gaining the reward.

Variable Structure Learning Automata (VSLA) [13] is represented by the quadruple (α, β, p, T) which, $\alpha = (\alpha_1, ..., \alpha_r)$ is a set of actions, $\beta = (\beta_1, ..., \beta_r)$ is a set of inputs, $p = (p_1, ..., p_r)$ is the probability vector in which each action may choose by it and $p(k+1) = T[\alpha(k), \beta(k), p(k)]$ is the learning algorithm. The following is a typical linear learning algorithm. Assume that the action α_i is chosen at time *k* as a sample realization from distribution (*k*). By considering $\beta \in \{0,1\}$, A linear schema for updating probability vector of LA with *r* actions is defined as (6) when $\beta = 0$ (favorable answer) and (7) when $\beta = 1$ (unfavorable answer).

$$\begin{aligned} & p_j(k+1) = p_j(k) + a[1 - p_j(k)] \quad i = j \\ & p_j(k+1) = (1-a)p_j(k); \qquad \forall j \;\; i = j \end{aligned} \quad (6)$$

$$\begin{aligned} & p_j(k+1) = (1-b)p_j(k) \qquad\qquad i = j \\ & p_j(k+1) = (b/r - 1) + (1-b)p_j(k) \;\; \forall j \;\; i \neq j \end{aligned} \quad (7)$$

Where *a*, *b* are respectively the reward and penalty parameters. By considering their values, they could have three states. When *a=b*, the algorithm named as the linear reward-penalty (L_{RP}). When *0<b<<a<1*, the algorithm named as $L_{R\varepsilon P}$ and When *b=0*, the algorithm named as L_{RI}. For more information about learning automata the reader may refer to [8].

B. PSO based on Learning Automata algorithms

From the family of PSO-LA model [10–12]. In [10] an intellectual and social movement algorithm based on learning automata has been proposed, in which one LA controls the behavior of the whole swarm. This automaton has two actions: ''*Follow the best*'' and ''*Continue your way*''. If the LA chooses ''*Follow the best*'' action, based on (8) best personal experience and best global experience will have role on updating the particle's velocity. In this case also the particles will ignore their current velocity. If the LA chooses ''*Continue your way*'' action, the current velocity of the particles will be set as the new velocity of them and particles will continue their current path.

$$v_{id}(k+1) = c_1 r_1 (p_{id}(k) - x_{id}(k)) + c_2 r_2 (p_{gd}(k) - x_{id}(k)) \quad (8)$$

The Learning Automaton [8], [13] has a number of actions which selects them stochastically. LA has a probability vector in which the probability of all actions is subscribed in it and LA does not aware of this probabilities. At first LA selects actions by chance, but in long time the aim is to take an action which gain the utmost reward.

In an attempt to reduce the probability of trapping PSO-LA into local minima, four new versions of PSO-LA model are introduced in [12], named as LAPSO1, LAPSO2, LAPSO3 and LAPSO4. LAPSO1 is similar to PSO-LA model introduced in [10]. Unlike [10], LAPSO2 uses one LA per each particle. The LA acts as the leader of the swarm and controls the particle's movement in the search space. Each LA has two actions: ''*Follow the best*'' and ''*Continue your way*''. If the LA chooses ''*Follow the best*'' action, with initial zero inertia weight, the particle moves in the search space toward the best position that meet by the whole swarm (*gbest*) and its best personal position (*pbest*). According to (8), this action performs local search. If the LA chooses ''*Continue your way*'' action, the particle will move with particular acceleration in the search space and will continue to move on its current path. According to (1), this action leads to global search and explore the unknown areas of space.

LAPSO3 is similar to LAPSO2; except for its way to reinitialize the particle's velocity while it is trapped into

local minima. Also LAPSO4 is similar to LAPSO2 except for the way, the reinforcement signal is computed. For more details about evaluation methods of PSO-LA model the reader may refer to [12].

IV. DYNAMIC GOBAL AND LOCAL COMBINED PSO BASED ON 3-ACTION LEARNING AUTOMATA (DPSOLA)

A. The proposed alterations and additions to the reviewed algorithms

In fact the trend of DGLCPSO algorithm [7] is similar to standard PSO [2]; however in DGLCPSO algorithm, an iterative dynamic global and local particle swarm optimization formula has been used (compare (1) with (3)). Simulation results show that DGLCPSO algorithm has faster convergence rate than standard PSO and also the algorithm shows better performance in large optimization problems. The DGLCPSO algorithm searches according to three strategies: (1) Individual knowledge of each particle which contains particle's best previous position and denoted as *pbest*. (2) Local knowledge of swarm's subgroup which is extracted from a specific neighborhood of particles, denoted as *lbest*. (3) Public knowledge of the whole swarm which is the best current solution, denoted as *gbest*. So, for updating particle's velocities, we can simultaneously use these three types of knowledge. The standard PSO just searches through the first and third strategies.

Using LA has two advantages: (1) using the existing knowledge for tuning the weight inertia or other adaptive parameters. (2) By getting feedback from the environment, trend will correct itself. In the proposed model, in each generation the LA controls the trend of particle's search by selecting an action from a finite set of actions. The LA tries to determine the optimal action, iteratively. Having one LA for the whole swarm may cause to lose the useful information of the particles [10]. Assigning one LA per each particle saves this useful information and leads to have diverse particles in the swarm. So, the proposed model reduces the probability of trapping in the local minima.

There is always a risk of trapping in local minima [10], [12].when the LA selects the ''*Follow the best*'' action and repeatedly gain reward, the particle performs a local search. If the particle converges to a point which is a local minima, after some iterations the LA will select the ''*Continue your way*'' action and particle's velocity will be set to zero. Since the velocity become zero the particle couldn't move and will trapped in that point. By combining the three noted search strategies, the proposed model will have better flexibility while the particle traps in local minima. The particle can use its local knowledge to escape from the local minima and eventually has higher exploration ability.

By reviewing the simulation results of introduced PSO-LA models in [12], the parameters which have been set for benchmark functions are so week, in contrast to our work. The maximum number of dimensions there, is 30 while in our algorithm, it is set to 150. The maximum number of generation there is 30000 while we have set it to 5000 for both unimodal and multimodal functions. Although DPSOLA run in much more realistic environment while still the proposed algorithm shows better results. Although DPSOLA run in much more realistic environment, the proposed algorithm shows better results.

B. Neighborhood definition

Since in DPSOLA algorithm, particles search through their local neighborhood and share their local information, we define three types of neighborhood topology for the proposed method. Having different types of neighborhood topology, let the particles to ''*fly freely*'' in the subspace of the problem. (1) **Fixed neighborhood (Fix):** In this topology the initial swarm considers as a vector and will slice to fixed pieces which contain specific number of particles. The particles neighborhood will not alter during the run. (2) **Ring neighborhood (Ring):** In this topology swarm consider as a circle and will slice by the specific radius. Similar to the previous topology the neighborhood is fixed during the run. (3) **Euclidean neighborhood (Ed):** In this topology the particles neighborhood will be defined dynamically according to the Euclidean distance between them. After calculating the fitness of all particles which place in the same neighborhood, the best fitness will consider as *lbest*.

C. DPSOLA algorithm

In order to obviate the cons of [2], [7], [10–12] in this paper we propose a new Dynamic Global and Local Combined Particle Swarm Optimization based on 3-action Learning Automata (DPSOLA) algorithm. For matching the structure of LA with DPSOLA, we use a three action learning automaton which can cover three aspects of proposed algorithm.

In DPSOLA, we assign one LA to each particle of the swarm. Each LA acts as the particle kernel and leads the particle's movement. Each LA has three actions: ''*Global and Local Combined Search*'', ''*Local Search*'', ''*Global Search*''. If the LA chooses ''*Global and Local Combined Search*'', according to (3), the particle will move with fixed acceleration and perform global and local search in the problem space. If the LA chooses ''*Local Search*'', according to (9), the particle will move toward the best position that meet by *lbest* with zero initial inertia weight. If the LA chooses ''*Global Search*'', according to (10) the particle will move toward the best position that meet by *gbest* with zero initial inertia weight.

$$v_{id}(k+1) = r_1(a+1/(endgen+1-k))*(p_{id}(k)-x_{id}(k)) + (b-1/(endgen+1-k))*(p_{ld}(k)-x_{id}(k)) \quad (9)$$

$$v_{id}(k+1) = r_1(a+1/(endgen+1-k))*(p_{id}(k)-x_{id}(k)) + cr_2(p_g(k)-x_{id}(k)) \quad (10)$$

Updating the velocity and position vectors of each particle, the reinforcement signal β will be generated according to (11) which are given below:

$$\beta = \begin{cases} 0 & if \quad fitness(X_i(k+1)) > fitness(X_i(k)) \\ 1 & otherwise \end{cases} \quad (11)$$

Evaluating the reinforcement signal, the probability vector of LA will be updated according to (6) or (7).

D. Pseudocode of DPSOLA

Algorithm 1 shows the Pseudocode algorithm for the DPSOLA, a PSO that splits the search method into 3 sub-methods. The probability vector of LA is updated at the end of each evaluation. This algorithm has the advantage that the condition checking performs in each evaluation.

Algorithm 1. DPSOLA

Step 1: Initialize parameters including: number of particles (*PS*), maximum generation number (*endgen*) and the other useful parameters.

Step 2: Scheduling and assignment

- Randomly generate initial population and particles velocity
- Calculate particles fitness from fitness function
- Initialize *gbest* position with the swarm's lowest fitness.
- Initialize *pbest* position with the current particle's components.
- Initialize *lbest* position with the best particle of particle's local neighborhood.
- Initialize Learning Automata parameters

while (do not reach the maximum generation (*endgen*))

- LA randomly selects an action and generate next swarm
- Evaluate swarm
  - Updating particle velocity
    - If LA selects ''Global and *Local Combined Search*'', then the velocity will update by (3).
    - If LA selects ''Local Search'', then the velocity will update by (9)
    - If LA selects ''*Global Search*'', then the velocity will update by (10)
  - Check the boundary and velocity conditions
  - Updating particle's position by (4)
  - Calculate new particle's fitness by fitness function
  - Check the boundary conditions
  - Calculate new values of *gbest*, *pbest* and *lbest* of each particle by comparison
  - Update *lbest* by (5)
  - Updating LA probability vector by (6),(7),(11)
- End of Evaluation

end of While.

Step 3: show results

V. Experimental Results and Analysis

A. Benchmark functions

In order to show the performance of DPSOLA algorithm, we use eight benchmark functions. The first half of Table I lists the function parameters. *f1-f4* are unimodal functions, *f5-f8* are multimodal functions with many local minima [2], [6], [7]. The reader should keep the following in mind: *ND* (Number of Dimensions), *FB* (Feasible Bounds). In all experiments the population size is set to 150. Each experiment runs for 5000 iteration per function. The mean of results, over 10 separate runs, are shown in tables II through IX.

B. Analysis of results

To evaluate the performance of DPSOLA algorithm we compare it with Original PSO and DGLCPSO algorithms [2], [7]. The proposed method is tested with both L_{RP} and $L_{R\varepsilon P}$ learning algorithms. The second half of table I lists the neighborhood parameters used to evaluate DPSOLA. The column ''*NT*'' denotes the Neighborhood Type and the column ''*NS*'' denotes the Neighborhood Size.

TABLE I.
Test Functions Parameters

Function Status			Topology Status			
			L_{RP}		$L_{R\varepsilon P}$	
Function	**ND**	**Range**	**NT**	**NS**	**NT**	**NS**
Sphere (f1)	150	[-100, 100]	Ring	5	Ring	5
Schwefel 1.2 (f2)	30	[-100, 100]	Ring	5	Ring	5
Schwefel 2.21 (f3)	30	[-100, 100]	Ed	5	Ring	5
Rosenbrock (f4)	25	[-100, 100]	Ring	7	Ring	7
Ackley (f5)	30	[-32, 32]	Ed	7	Ed	7
Griewank (f6)	150	[-600, 600]	Ring	7	Ring	7
Penalized P8 (f7)	100	[-50, 50]	Ring	7	Fix	7
Penalized P16 (f8)	100	[-50, 50]	Ring	7	Fix	7

In all experiments acceleration constants (*c, c_1 and c_2)* are set to *2*. The following format used in tables II through IX. The first column lists the inertia weight (*w*) values which consider as a constant. The third column lists weights index (*a, b*). L_{RP} and $L_{R\varepsilon P}$ in the two last columns denote as LA learning algorithm which is used in DPSOLA. In L_{RP} learning algorithm, *alpha=beta=0.01* and in $L_{R\varepsilon P}$ *alpha=0.001 and beta = 0.01*. The boundary condition is set to the whole feasible bounds. Since DPSOLA is implemented as a full parametric algorithm, the trend of performing the experiments is as follow: Because of the large number of feasible experiments for DPSOLA, first we run every possible combination of parameters and then choose the best fitness of them. For each test function, we run this optimal solution for 10 times and the average of these solutions is reported in II through IX. The optimum solutions of proposed method were marked with bold letters.

TABLE II.
Sphere (f_1) after 10 Times of 5000 Evaluations

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	4.07E+05	1.05	2.80E+00	6.91E-18	1.47E-01
		1.005	5.90E-03	7.35E-03	1.84E-01
		1.0005	2.40E-03	1.48E-07	2.07E-01
0.2	3.92E+05	1.05	9.54E-04	1.82E-02	1.34E-01
		1.005	4.96E-02	1.79E-02	1.17E-01
		1.0005	5.34E-01	2.50E-02	8.17E-02
0.3	3.91E+05	1.05	2.83E-02	1.87E-25	5.91E-02
		1.005	7.80E+00	5.67E-04	7.36E-02
		1.0005	1.61E-01	3.87E-87	1.53E-01
0.4	3.88E+05	1.05	3.90E-03	1.41E-42	1.20E-01
		1.005	3.66E-02	3.76E-05	6.36E-02
		1.0005	3.48E-02	**1.38E-212**	9.33E-02
0.5	3.99E+05	1.05	8.06E-01	1.12E-122	4.76E-02
		1.005	9.04E-02	5.42E-172	2.47E-02
		1.0005	6.86E-02	5.61E-102	1.04E-01

TABLE III.
Schwefel 1.2 (f_2) after 10 Times of 5000 Evaluations

W	OPSO	a, b	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	8.36E+01	1.05	3.84E-01	2.62E-03	1.33E+01
		1.005	5.09E-04	2.90E-03	2.77E+02
		1.0005	1.27E-04	7.26E-04	1.16E+03
0.2	1.24E+00	1.05	1.10E-05	1.30E-03	1.68E-03
		1.005	1.15E-08	4.46E-03	1.09E+02
		1.0005	4.77E-09	2.84E-03	1.98E+02
0.3	6.67E-02	1.05	4.52E-07	8.26E-04	2.80E-03
		1.005	4.94E-10	1.47E-36	7.47E-03
		1.0005	1.36E-01	7.62E-04	2.27E+01
0.4	5.67E-01	1.05	3.33E-06	1.34E-03	1.15E+02
		1.005	4.16E-09	3.93E-04	1.94E+02
		1.0005	1.28E-09	8.09E-04	6.26E+01
0.5	9.32E+00	1.05	8.10E-03	7.21E-04	5.09E+01
		1.005	3.61E-05	1.72E-172	6.33E+01
		1.0005	4.48E-06	**7.47E-175**	3.33E-03

TABLE IV.
SCHWEFEL 2.21 (f_3) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	9.41E+00	1.05	6.07E-02	3.85E-11	2.73E-07
		1.005	8.08E-04	1.26E-10	1.49E-05
		1.0005	5.87E-04	1.60E-10	4.72E-05
0.2	5.25E-01	1.05	3.40E-05	1.37E-11	8.63E-09
		1.005	1.94E-06	3.55E-11	1.38E-07
		1.0005	5.52E-07	8.55E-11	4.13E-07
0.3	2.92E-02	1.05	1.16E-06	6.37E-13	2.42E-10
		1.005	8.09E-08	1.76E-12	5.49E-09
		1.0005	4.35E-08	1.04E-10	1.34E-08
0.4	8.21E-02	1.05	6.85E-07	8.75E-13	1.88E-12
		1.005	5.07E-08	1.49E-11	3.48E-11
		1.0005	6.06E-08	5.00E-12	1.82E-10
0.5	9.89E-01	1.05	5.13E-05	2.63E-12	**3.18E-13**
		1.005	1.36E-06	1.85E-12	1.34E-12
		1.0005	5.61E-07	2.83E-12	4.05E-13

TABLE V.
ROSENBROCK (f_4) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	7.61E+01	1.05	7.09E+01	1.41E+00	7.60E-01
		1.005	1.42E+01	7.66E-01	3.46E-01
		1.0005	7.75E+00	1.38E+00	1.11E+00
0.2	1.35E+02	1.05	1.43E+02	7.40E-01	1.00E+00
		1.005	5.45E+01	4.26E-01	7.53E-01
		1.0005	2.77E+00	7.66E-01	5.55E-01
0.3	9.69E+01	1.05	6.92E+00	1.55E+00	3.90E-01
		1.005	2.92E+01	9.12E-01	1.05E+00
		1.0005	8.95E+00	4.91E-01	1.24E+00
0.4	1.20E+01	1.05	5.54E+00	**2.76E-01**	1.07E+00
		1.005	2.36E+01	7.41E-01	2.75E-01
		1.0005	4.43E+01	3.72E-01	1.34E+00
0.5	1.41E+02	1.05	2.70E+02	9.20E-01	2.96E-01
		1.005	1.25E+02	4.47E-01	6.98E-01
		1.0005	9.74E+00	3.83E-01	6.98E+00

TABLE VI.
ACKLEY (f_5) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	5.49E-05	1.05	7.28E-15	1.72E-14	1.58E-14
		1.005	5.15E-15	2.43E-14	**2.54E-14**
		1.0005	5.86E-15	2.72E-14	2.33E-14
0.2	6.88E-01	1.05	5.51E-15	1.72E-14	1.87E-14
		1.005	2.84E-01	2.50E-14	1.55E-14
		1.0005	5.86E-15	2.97E-14	2.15E-14
0.3	9.31E-02	1.05	6.22E-15	1.97E-14	1.87E-14
		1.005	1.03E-01	2.61E-14	1.83E-14
		1.0005	4.80E-15	2.86E-14	2.18E-14
0.4	7.64E-15	1.05	5.51E-15	1.97E-14	1.69E-14
		1.005	4.80E-15	2.75E-14	1.94E-14
		1.0005	4.80E-15	2.22E-14	2.36E-14
0.5	9.41E-15	1.05	6.93E-15	2.26E-14	2.43E-14
		1.005	5.51E-15	3.29E-14	1.83E-14
		1.0005	5.51E-15	3.07E-14	2.65E-14

TABLE VII.
GRIEWANK (f_6) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	3.72E+03	1.05	4.52E-01	3.19E+00	6.01E+01
		1.005	1.26E-02	1.20E+01	6.14E+01
		1.0005	6.45E-02	9.82E+00	4.91E+01
	3.84E+03	1.05	1.14E-01	1.23E+01	5.02E+01
		1.005	6.12E-01	1.35E+01	6.21E+01
0.2		1.0005	2.77E-01	2.72E+00	6.37E+01
0.3	3.58E+03	1.05	6.14E-02	3.87E+00	6.65E+01
		1.005	2.65E-01	4.89E+00	6.62E+01
		1.0005	2.07E-01	6.68E+00	6.00E+01
0.4	3.27E+03	1.05	3.75E-02	2.69E-01	6.94E+01
		1.005	2.18E-01	4.36E+00	6.44E+01
		1.0005	6.66E-02	7.12E+00	7.34E+01
0.5	3.67E+03	1.05	2.78E-01	**2.29E-03**	5.43E+01
		1.005	6.41E-02	9.88E+00	5.43E+01
		1.0005	5.83E-02	1.46E+01	5.43E+01

TABLE VIII.
PENALIZED P8 (f_7) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	*a, b*	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	2.52E+09	1.05	8.54E+07	9.65E-02	1.24E-02
		1.005	5.25E-02	4.07E-01	1.25E-02
		1.0005	9.70E-03	1.25E-01	6.24E-03
0.2	2.45E+09	1.05	3.12E-02	9.03E-02	9.34E-03
		1.005	3.11E-02	9.64E-02	2.18E-02
		1.0005	3.12E-02	5.60E-02	1.24E-02
0.3	2.36E+09	1.05	4.05E-02	9.96E-02	2.80E-02
		1.005	6.54E-02	1.31E-01	3.11E-03
		1.0005	2.18E-02	4.67E-02	1.24E-02
0.4	2.29E+09	1.05	4.07E-02	3.42E-02	6.22E-03
		1.005	1.12E-01	1.53E-01	2.18E-02
		1.0005	3.73E-02	1.21E-01	**9.33E-03**
0.5	2.27E+09	1.05	3.97E+01	8.09E-02	3.11E-02
		1.005	8.69E-02	8.09E-02	1.56E-02
		1.0005	5.84E-02	3.42E-02	1.87E-02

TABLE IX.
PENALIZED P16 (f_8) AFTER 10 TIMES OF 5000 EVALUATIONS

W	OPSO	a, b	DGLCPSO	$DPSOL_{RP}$	$DPSOL_{R\varepsilon P}$
0.1	2.46E+09	1.05	1.60E+08	2.61E-01	6.76E-02
		1.005	7.01E-04	2.04E-01	4.69E-02
		1.0005	2.34E-04	2.84E-01	1.75E-05
0.2	2.38E+09	1.05	2.70E-04	1.26E-01	5.52E-02
		1.005	1.97E-05	1.24E-01	1.81E-02
		1.0005	1.20E-04	1.04E+00	2.72E-02
0.3	2.24E+09	1.05	5.13E-02	8.48E-02	1.81E-02
		1.005	3.51E-06	1.63E-01	1.81E-02
		1.0005	2.00E-02	2.93E-01	3.71E-02
0.4	2.43E+09	1.05	4.00E-02	2.62E-01	3.62E-02
		1.005	6.98E-05	2.90E-01	9.06E-03
		1.0005	4.20E-05	2.73E-01	3.70E-02
0.5	2.24E+09	1.05	2.61E+01	1.35E-01	3.79E-02
		1.005	5.67E-02	9.55E-02	**4.27E-12**
		1.0005	4.21E-02	3.79E-02	6.67E-02

All swarms in eight PSO algorithm initialized in an area equal to primary feasible bounds in each dimension, this kind of initialization provide a realistic test bed. Having a look on table I, One can understand that in unimodal functions, *Ring* neighborhood topology with 5 particles in it, has the best fitness on the benchmark functions. The first three unimodal functions (f_1-f_3) are easily optimized by the DPSOLA, with the 5 particles in each local neighborhood. f_4 needs a little more particles (7) to perform as well as DGLCPSO. In order to have better results and keep the balance between subswarms, in all multimodal functions (f_5-f_8), we put 7 particles in each local neighborhood of the swarm. Again *Ring* neighborhood remains as the majority choice.

By overall observation on tables II through V, there are a very large difference in performance between the L_{RP} and $L_{R\varepsilon P}$ learning algorithms. In all unimodal functions L_{RP} learning algorithm with *alpha, beta = 0.01* performs better. Using L_{RP} learning algorithm lets the LA to select the best action for its current path. In f_1-f_3, DPSOLA algorithms show significantly better performance. Having a look on tables VI through IX, in multimodal functions (*f5-f8*), the proposed algorithms again suppress both DGLCPSO and PSO. Except for Ackley's function that the mean fitness of DPSOLA cannot exactly reach DGLCPSO but the answer is still reasonable. In multimodal functions the overall performance of $L_{R\varepsilon P}$ is better than L_{RP} learning algorithm.

From tables II through IX one can understand that the best solutions are for $w \in (0.4, 0.5)$ and especially $w = 0.4$ offers better performance. The best values for a, b coefficients are $a, b = 1.0005$. The best embedded learning automaton is L_{RP} for unimodal functions and $L_{R\varepsilon P}$ for multimodal functions. By comparing these results, it can be clearly perceived that DPSOLA always has better solution than PSO and DGLCPSO, in both unimodal and multimodal functions. Although we discussed about the variation effects of a, b coefficients, there still remain some area of open search, which will be consider later.

Convergence graph of DGLCPSO, DPSOLA, PSO and PSOLA algorithms on Penalized P16 function is shown in figure 1. The standard PSO and PSOLA algorithms trapped in local minima, as can be seen from the flat aggregated lines in figure 8. The $DPSOL_{RP}$ manage to continue but trapped in local minima. The $DPSOL_{R\varepsilon P}$ algorithm is able to continue its solution and offer the best performance.

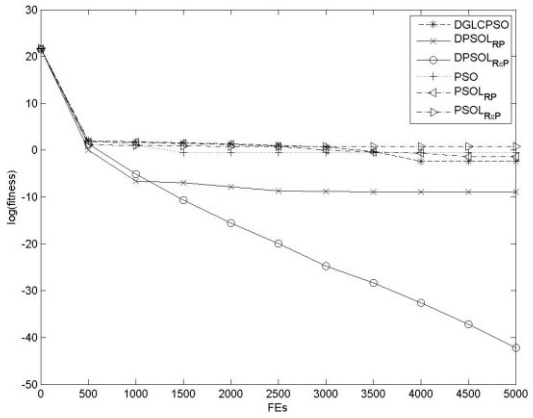

Figure 1. Penalized P16 (*f8*) Convergence graph

VI. CONCLUSION

In this paper we introduced a new algorithm from the family of PSO-LA model named as DPSOLA and study its internal mechanism. Standard PSO [2], DGLCPSO [7] and previous PSO-LA models [10–12] always have the problem of searching around local minima, now by a set of three search strategies introduced by DPSOLA, we suggest a scapegoat solution. DPSOLA splits the swarm into the subswarms in which the particles could share their useful information. The LA takes the responsibility of moving each particle in problem space. The proposed algorithm tested in eight benchmark functions with large scale swarm size, high dimension, sufficient number of iterations, equal initialization range to feasible range and a few particles in local neighborhood (subswarms). These intensive conditions make the experiments so realistic. The approach performs better and better by increasing the inertia weight. Results show that the proposed method reduces the probability of trapping in local minima and obtains the optimum fitness. Also the 3 action learning automaton embedded in DPSOLA makes the convergence process faster. Keeping the number of subswarms members lowly noticeable (considering 5 or 7 in the current experimental setup), leads the subswarms particles to the best fitness they can reach in unimodal and multimodal problems in contrast with other algorithms.

REFERENCES

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

[2] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in *Proceedings of IEEE Swarm Intelligence Symposium (SIS)*, pp. 120-127, April 2007.

[3] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1205-1213, June 2007.

[4] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC)*, vol. 3, pp. 1945-1950, July 1999.

[5] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658-3670, June 2011.

[6] F. van den Bergh and A. P. Engelbrecht, "A Cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225- 239, June 2004.

[7] B. Jiao, Z. Lian, and Q. Chen, "A dynamic global and local combined particle swarm optimization algorithm," *Chaos, Solitons & Fractals*, vol. 42, no. 5, pp. 2688-2695, December 2009.

[8] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*. Prentice-Hall Inc, 1989.

[9] 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 (ICMLA),* pp. 456- 462, December 2004.

[10] M. Sheybani and M. R. Meybodi, "PSO-LA: A New Model for Optimization," in *Proceedings of 12th Annual CSI Computer Conference of Iran*, pp. 1162-1169, February 2007.

[11] B. Jafarpour, M. R. Meybodi, and S. Shiry, "A hybrid method for optimization (discrete PSO + CLA)," in *International Conference on Intelligent and Advanced Systems,* pp. 55-60, November 2007.

[12] M. Hamidi and M. R. Meybodi, "New Learning Automata based Particle Swarm Optimization Algorithms," in *Proceedings of the 2nd Iranian Data Mining Conference (IDMC)*, pp. 1-15, November 2008.

[13] A. B. Hashemi and M. R. Meybodi, "A note on the learning automata based algorithms for adaptive parameter selection in PSO," *Applied Soft Computing*, vol. 11, no. 1, pp. 689-705, January 2011.

[14] A. B. Hashemi and M. R. Meybodi, "Adaptive parameter selection scheme for PSO: A learning automata approach," in *Proceedings of 14th International CSI Computer Conference (CSICC),* pp. 403-411, October 2009.