# A Robust Heuristic Algorithm for Cooperative Particle Swarm Optimizer: A Learning Automata Approach

Mohammad Hasanzadeh, Mohammad Reza Meybodi and Mohammad Mehdi Ebadzadeh
Computer Engineering and Information Technology Department
Amirkabir University of Technology (Tehran Polytechnic)
Tehran, Iran
mdhassanzd@aut.ac.ir, mmeybodi@aut.ac.ir, ebadzadeh@aut.ac.ir

*Abstract*— **This paper presents a modification of Particle Swarm Optimization (PSO) technique based on cooperative behavior of swarms and learning ability of an automaton. This approach called the Cooperative Particle Swarm Optimization based on Learning Automata (CPSOLA). The CPSOLA algorithm uses three-layer cooperation: intra swarm, inter swarm and inter population. There are two active populations in CPSOLA. In the primary population, the particles are placed in all swarms and each swarm consist of multiple dimensions of search space. Also there is a secondary population in CPSOLA which is used the conventional PSO's updating format. In the upper layer of cooperation, the embedded Learning Automaton (LA) is responsible for deciding whether to cooperate between populations or not. Experiments are organized on five benchmark functions and results show notable performance and robustness of CPSOLA, cooperative behavior of swarms and successful adaptive control of populations.**

***Keywords- Particle Swarm Optimization (PSO); Learning Automata (LA); Cooperative learning.***

## I. INTRODUCTION

Particle Swarm Optimization (PSO) [1], [2] is a population based technique inspired form shoaling behavior of fish and swarming behavior of insects. The mystery becomes evident when the simple rules that followed by individuals leads to *emergent* of well-organized system. Cooperative PSO (CPSO) [3] is a variation of the traditional PSO algorithm in which the dimensions of population divided into multiple separate swarms and each swarm try to optimize the problem separately. During the fitness evaluation of the particles, the cooperation is occurred between swarms. Comprehensive Learning PSO (CLPSO) [4] is one of the most successful PSO improvements. A new learning strategy is used in CLPSO, where all particles' best information is used to update any other particle's velocity. The *inertia weight* $(w)$ [5] is one of the most important PSO's parameters, which is used to balance the global and local search of the population. Recently, an Adaptive PSO (APSO) [6] has introduced. APSO adaptively controls the PSO parameters by estimating the population distribution. Beside the adaptation of the *inertia weight*, APSO algorithm controls *acceleration coefficients* $(c_1 \text{ and } c_2)$ by four strategies named as exploration, exploitation, convergence and jumping out.

A Learning Automaton (LA) [7], [8] is a machine which is adapted to changes in its environment. The adaption is the result of *learning* process of the automaton. Recently learning automata is used for adaptive parameter selection in Evolutionary Algorithms (EA) [9], [10]. Also a new hybrid method of optimization which called PSO-LA [11–13] has been emerged. In PSO-LA algorithms an LA or a group of LAs is assigned to the whole population or each particle of the population. LA or group of LAs controls the path and velocity of the particles.

CPSO family [3] consists of four algorithms: CPSO-S, CPSO-$S_K$, CPSO-H and CPSO-$H_K$. K is the *split factor* parameter which specifies the length of desired solution vector. Typically, while optimizing an N – dimensional problem by using CPSO-S, K will be set to N (number of dimensions). Having both beneficial characteristics of PSO and CPSO-$S_K$, CPSO-$H_K$ is a combination of these two algorithms. It is a tempting idea to have a mechanism which is able to understand when to switch between PSO and CPSO-$S_K$ [3]. In this paper we significantly improve this hybridization by embedding an automaton as a tollman of the switching mechanism.

The paper is organized as follow: section 2 reviews the standard PSO and Cooperative PSO. Section 3 introduces learning automaton and its application in PSO. Section 4 describes cooperative PSO based on learning automata. Experimental setup and simulation results are presented in section 5.

## II. PARTICLE SWARM OPTIMIZATION (PSO)

### A. Conventional Particle Swarm Optimization

Particle Swarm Optimization (PSO) [1], [2] consists of a population of particles in which each particle represents a feasible solution vector. Assume that we have an N – dimensional problem space with M particles which is initialized in the feasible search boundary. The velocity, position and the best previous position of the *i*th particle are respectively shown by $X_i = \left(x_i^1, x_i^2, ..., x_i^N\right)$, $V_i = \left(v_i^1, v_i^2, ..., v_i^N\right)$ and $pbest_i = \left(pbest_i^1, pbest_i^2, ..., pbest_i^N\right)$. Also the best position of the population is $gbest = \left(gbest^1, gbest^2, ..., gbest^N\right)$. The velocity $V_i^d$ and position $X_i^d$ of the *d*th dimension of the *i*th particle are manipulated through the following equations [4]:

$$V_i^d = w \times V_i^d + c_1 \times rand1_i^d \times \left(pbest_i^d - X_i^d\right) + c_2 \times rand2_i^d \times \left(gbest^d - X_i^d\right) \quad (1)$$

$$X_i^d = X_i^d + V_i^d \quad (2)$$

Where $c_1$ and $c_2$ are *acceleration constants* which absorb the particles to *pbest* and *gbest* positions. $w \in [0,1]$ is *inertia weight* which controls the global and local search. *rand1* and *rand2* $\in [0,1]$ are two random numbers generated for each dimension of the particles.

### B. Cooperative Learning

The idea of cooperative learning was first implemented in the field of GA by Potter [14]. Potter suggested that for optimizing the designated target function, each dimension of the fitness function could be optimized by a distinct population and be evaluated in form of an N – dimensional vector through the fitness function. PSO and GA both suffer from the "curse of dimensionality". Using potter's aforementioned technique in PSO must leads to have promising results in function optimization. Recently The concept of cooperation mapped into PSO technique. Cooperative behavior in PSO was first introduced by Van den Bergh [3], [15]. In Cooperative PSO instead of having one swarm of M particles trying to optimize the designated N – dimensional optimization problem, we have N swarms of M particles which working on an isolated 1 – dimensional problem. In this approach we should use a *context vector* to build a required N – dimensional vector to evaluate each of the swarms.

The family of CPSO algorithm proposed in [3] consists of the following algorithms: CPSO-S, CPSO-$S_K$, CPSO-H and CPSO-$H_K$. In CPSO-S algorithm each dimension of search space is considered as a swarm of M particles and all of the swarms are trying to find a better solution vector in each iteration. If there is any correlation in the population, it would be desirable to gather the correlated dimensions in the same swarm. The idea of correlated variables leads to emergence of *split factor* parameter which tuned the swarm size. Now, instead of splitting the population into N swarm of 1 - dimensional vectors like CPSO-S, we could have K swarms of C – dimensional vectors (C<N) like CPSO-$S_K$. Standard PSO algorithm has the ability of escaping from local minima and CPSO-$S_K$ algorithm has fast convergence speed. Merging both beneficial characteristic of this two algorithms leads to appearance of CPSO-$H_K$ algorithm. CPSO-$H_K$ algorithm consists of two phases, in the first phase CPSO-$S_K$ algorithm run and the information exchange *from* CPSO-$S_K$ half *to* PSO half of the algorithm performs. At the second phase PSO algorithm run and the information exchange *form* PSO half *to* CPSO-$S_K$ half performs. Note that each phase performs in a separate iteration.

## III. Learning Automata (LA)

### A. Learning Automata (LA) Scheme

Learning Automata (LA) [7], [8] is a stochastic optimization technique from the family of Reinforcement Learning (RL) algorithms. Having enough interaction with the unknown environment, elegance emerges and the optimal policy will be chosen. Fig. 1 shows how automaton interacts with its environment. A study of the learning process of LA in a random environment is comprehensively reported in [7].

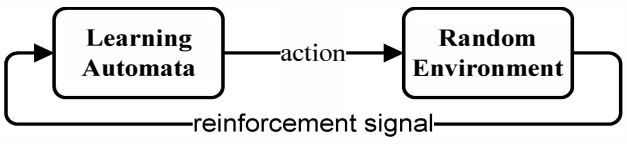


Figure 1. The interaction between learning automata and environment.

Learning automatons [7], [8] are divided into two groups which are *fixed-structure automaton*s and *variable-structure automatons*. A Variable-Structure Learning Automaton (VSLA) is represented by a quadruple $[\alpha, \beta, p, T]$, Where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is a set of actions, $\beta = \{\beta_1, \beta_2, ..., \beta_r\}$ is a set of inputs, $p = \{p_1, p_2, ..., p_r\}$ is the probability vector corresponds to each action and $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ is the learning algorithm. If *p(n+1)* is a linear function of *p(n)* then the reinforcement scheme should be linear; otherwise it is nonlinear. In the simplest form of VSLA consider an automaton with *r* actions in a stationary environment where $\beta = \{0,1\}$ is included in inputs. After selecting the action by the automaton, the reinforcement signal will receive from the environment. When the *positive response* $(\beta = 0)$ received form the environment, the action probabilities are updated through Equation (3):

$$p_j(n+1) = \begin{cases} p_j(n) + a.(1 - p_j(n)) & \text{if } i = j \\ p_j(n).(1-a) & \text{if } i \neq j \end{cases} \quad (3)$$

When the *negative response* $(\beta = 1)$ received from the environment, the action probabilities are updated through Equation (4):

$$p_j(n+1) = \begin{cases} p_j(n).(1-b) & \text{if } i = j \\ \frac{b}{r-1} + (1-b).p_j(n) & \text{if } i \neq j \end{cases} \quad (4)$$

The *a* and *b* are called *learning parameters* and they are associated with the reward and penalty responses. If *a* and *b* are equal, the learning scheme is called $L_{R-P}$ (Linear Reward-Penalty). If the learning parameter *b* is set to 0, then the learning scheme is named $L_{R-I}$ (Linear Reward-Inaction). And finally if the learning parameter *b* is much smaller than *a*, the learning scheme is called $L_{R\varepsilon P}$ (Linear Reward-epsilon-Penalty).

### B. Learning Automata based Particle Swarm Optimizer algorithms

Parameter adaption [9], [10] is one of the most difficult tasks in evolutionary algorithms. As there are multiple parameters in PSO, it needs a mechanism to tune them during the evaluation of the population. In [9] a study of adaptive PSO parameter selection is conducted. Embedding learning automata in the population of PSO is another improvement; the model is called PSO-LA. In PSO-LA model an automaton is used to configure the search behavior of particles and adjust the

velocity and position of them based on optimal selected policy. In coarse-grained PSO-LA [11] algorithms, an LA takes the responsibility of steering the whole swarm $\left(\left|population\right| = \left|LA\right|\right)$. Since coarse-grained PSO-LA get trapped into local minima, in fine-grained PSO-LA algorithms [12], [13], LA assigned to each particle of the swarm $\left(\left|population\,size\right| = \left|LA\right|\right)$. For more details about PSO-LA model readers may refer to [11–13].

## IV. Cooperative Particle Swarm Optimization based on Learning Automata (CPSOLA)

### A. Defining Scenarios

The CPSO model consists of four algorithms, from now on, our study specifically focused on CPSO-$H_K$ algorithm which covers the other three ones. Fig. 2 shows the structure of CPSO-$H_K$ algorithm. "When to switch between CPSO-$S_K$ and PSO?" is a question proposed by Van den Bergh in [3]. For designing such a robust, general and adaptive mechanism consider the following scenarios in CPSO-$H_K$ algorithm:

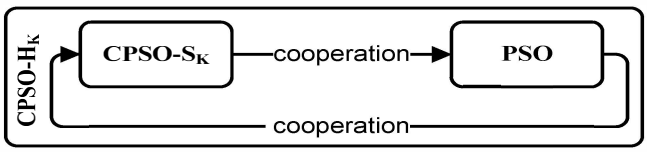


Figure 2. Structural view of CPSO-$H_K$ algorithm.

*Scenario 1*: At the earliest generations, the particles are scattered in the search space. It is desired to have fast global search of CPSO-$S_K$ while any local minima are placed around the particles. At the middle iterations of the algorithm, while getting trapped in a local minimum, using PSO is much beneficial to escape from it.

*Scenario 2:* Immediately after escaping from local minima, it is safe for the algorithm to make use the high speed of CPSO-$S_K$ till reaching the next local minimum.

*Scenario 3:* Sometimes the information exchange that is occurred in each generation of CPSO-$H_K$ algorithm is unnecessary. This unnecessary amount of cooperation defects the run time performance of the CPSO-$H_K$.

Reviewing the discussed scenarios, interleave execution of CPSO-$S_K$ and PSO seems to be a naïve form of cooperation between these two algorithms. A proper choice is to form an adaptive cooperation between CPSO-$S_K$ and PSO algorithms. By using one learning automaton, we could have an adaptive switching mechanism which intelligently switches between CPSO-$S_K$ and PSO algorithms. As well as preserving the positive characteristics of CPSO-$S_K$ and PSO algorithms, CPSOLA algorithm significantly reduce the amount of information exchange.

### B. Description of CPSOLA Algorithm

Like CPSO-$H_K$ in CPSOLA, we have two separate populations. The CPSO-$S_K$ population is our primary population and the PSO population is the secondary one. Information exchange between two populations is postponed to *critical generations* because there is an adaptive switching mechanism between these two algorithms. A critical generation is a part of evolution process in which the cooperation between CPSO-$S_K$ algorithm and PSO is vital. The scheme of adaptive switching mechanism of CPSOLA is shown in Fig. 3. The automaton has two actions: 1) Cooperation between primary and secondary population. 2) Isolation and just using primary population. In other words: 1) Running CPSO-$H_K$ algorithm. 2) Running CPSO-$S_K$ algorithm.

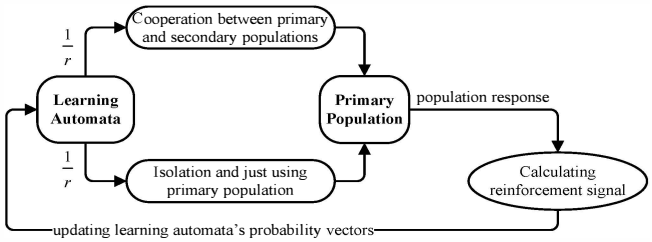


Figure 3. Schematic view of decision making by learning automata

In each generation of CPSOLA algorithm, automaton decides whether to start the alternative population (PSO population) or not. While having enough interaction with the populations, the automaton perceives when to switch between algorithms. We listed the five main differences between the CPSOLA and CPSO model: 1) Instead of having unnecessary information exchange in each generation, preserve it for *critical generations*. 2) Reduced the workload of algorithm and made the execution time faster. 3) Preserve CPSO-$S_K$ fast convergence speed property. 4) Keep the PSO's ability to escape from local minima. 5) Preserve the diversity of the population.

There is a criterion to evaluate the reinforcement signal in CPSOLA algorithm (Equation (5)). *If* in the current iteration global best position of primary population (CPSO-$S_K$ swarm best particle) improved *then* the automata's selected action would get the award and the automaton will be punished *otherwise*. Since the reinforcement signal is calculated in the context of primary population, the global best position of secondary population (PSO global best particle) do not have a direct influence on evaluating this signal.

$$\beta = \begin{cases} 0 & if\ \ fitness\left(Sbest_i < Sbest_{i-1}\right) \\ 1 & Otherwise \end{cases} \tag{5}$$

Fig. 4 is the pseudo code of CPSOLA, which an LA determines the time required to perform cooperation between primary and secondary populations.

### C. Analysing the Adaptive Cooperative Behavior

Learning automata's action selection needs a comprehensive perception of the environment. In this section, a 30 – dimensional Rosenbrock test function with 20 particles is used to investigate the interaction between LA and population during the evolution. The fitness comparison of CPSO-$S_K$ and PSO are plotted in Fig. 5.a. By looking on zoomed boxes which are taken from specific parts of the evolution, we can observe that:

**Algorithm 1** Cooperative PSO based on LA

**define**
  initialize primary population with K swarms: $P$
  initialize secondary population: $Q$
  initialize LA with 2 actions: {cooperation, isolation}
**do**
  select an action
  **if** the selected action is *cooperation* **then**
    **for** each swarm $P_j; j \in [1..K]$
      **for** each particle $i \in [1..s]$
        update particle position by Equations (1) & (2)
        calculate particle fitness
        update *pbest* & *gbest*
      **endfor**
    **endfor**
    select a random particle form $Q$ to write
    **for** each particle $Q; j \in [1..K]$
      update particle position by Equations (1) & (2)
      calculate particle fitness
      update *pbest* & *gbest*
    **endfor**
    **for** swarm $P_j; j \in [1..K]$
      select a random particle form $P$ to write
    **endfor**
  **elseif** the selected action is *isolation* **then**
    **for** each swarm $P_j; j \in [1..K]$
      **for** each particle $i \in [1..s]$
        update particle position by Equations (1) & (2)
        calculate particle fitness
        update *pbest* & *gbest*
      **endfor**
    **endfor**
  evaluate reinforcement signal by Equation (5)
  update LA's probability vectors by Equations (3) & (4)
**until** a terminate condition is met

Figure 4. The pseudo code of CPSOLA algorithm.

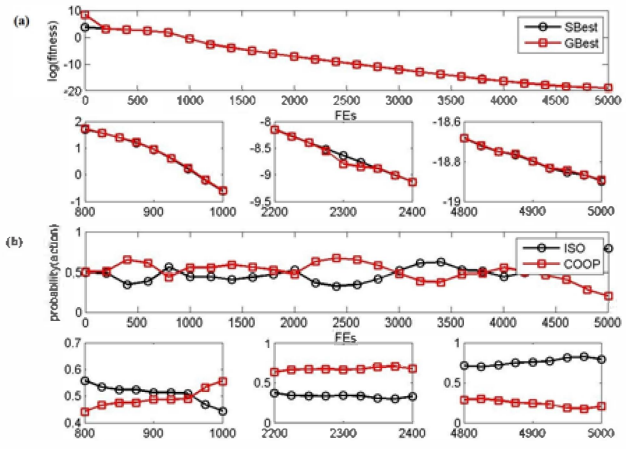


Figure 5. (a) CPSO-$S_K$'s *sbest* VS. PSO's *gbest*. (b) Learning automata's action selection variance.

In the earliest iterations of CPSOLA algorithm, there is no obvious distinct between the fitness of two algorithms. The algorithm should escape from the local minima after reaching the middle iteration, so the policy is changed to cooperation and during this part of evolution the PSO's fitness even could be better than CPSO-$S_K$'s fitness. Because of its cooperative multi – dimensional search method, in the last generations of running CPSOLA, the CPSO-$S_K$'s fitness becomes dominated and the algorithm converges faster.

The variance of action probabilities are plotted in Fig. 5.b. *Isolation* action means the algorithm is just used its primary population, hence the *Cooperation* action means two populations perform information exchange. It can be seen that CPSOLA algorithm has the ability to jump out of the local optima, which is the result of hidden diversity of PSO algorithm. Although the first and the second population perform cooperation but, while the learning automaton selects the *isolation* action, the PSO population skips some of the iterations. Since the cooperation dose not perform in each iteration, this trend seems to be a little unwanted. But by looking from the outer layer of cooperation, writing a bad fitness from alternative population into primary one could increase the diversity of primary population. In the middle generations, while primary population is stagnated in a local minimum, CPSOLA algorithm starts exchanging the information between two populations. This means that in a few generations, the secondary population could overwrite its inferior solutions to the primary population except the global best particle of each swarm which is protected. The diversity of primary population will increase significantly and the algorithm could escape from the local minimum.

## V. Experimental Results

### A. Simulation Setup

In order to have a normal comparison, all the PSO algorithms should use a same number of fitness. This number is set to $2.0\times10^4$ . All experiments were run 10 times; the means and variances of best solution of these runs are reported. In order to study the impact of population size; the experiments repeated with 10, 15 and 20 particles per population. All benchmark functions are 30 – dimensional optimization problems. The simulation results of the these test functions are presented in Table I.

Observing the learning automaton's [9], [10] behavior during the evolution process, three different kind of learning algorithms are placed in CPSOLA algorithm. The detailed configurations of the reward and penalty parameters are mentioned in Table II.

In order to compare the proposed method, we simulate four other PSOs, as detailed in Table III. We choose five benchmark functions [3], [4] to run the experimental tests. Rosenbrock and Quadric functions are unimodal and simple problems. Ackley, Rastrigin and Griewank functions are highly multimodal problems with many local optima positioned in their grid.

TABLE I. Results after $2.0 \times 10^4$ Fitness Evaluations

| *Algorithm/Func.* | *S* | *Rosenbrock ($f_0$)* | *Quadric ($f_1$)* | *Ackley ($f_2$)* | *Rastrigin ($f_3$)* | *Griewank ($f_4$)* |
|---|---|---|---|---|---|---|
| **PSO** | 10 | 1.30E-01 ± 1.45E-01 | 1.08E+00 ± 1.41E+00 | 7.33E+00 ± 6.23E-01 | 8.27E+01 ± 5.64E+00 | 9.65E-01 ± 7.58E-01 |
| | 15 | 5.53E-03 ± 6.19E-03 | 2.85E-72 ± 5.41E-72 | 4.92E+00 ± 5.81E-01 | 7.44E+01 ± 5.66E+00 | 2.62E-01 ± 1.61E-01 |
| | 20 | 9.65E-03 ± 7.28E-03 | 2.17E-98 ± 4.20E-98 | 3.57E+00 ± 4.58E-01 | 6.79E+01 ± 4.84E+00 | 6.51E-02 ± 2.17E-02 |
| **CLPSO** | 10 | 5.12E+00 ± 3.23E+00 | 2.96E+02 ± 1.78E+02 | 6.45E+00 ± 1.42E+00 | 1.74E+01 ± 4.60E+00 | 7.27E-01 ± 1.28E+00 |
| | 15 | 2.22E+00 ± 1.04E+00 | 9.79E+01 ± 6.98E+01 | 3.30E+00 ± 1.37E+00 | 7.26E+00 ± 2.85E+00 | 1.62E-02 ± 3.07E-02 |
| | 20 | 1.88E+00 ± 3.26E-01 | 4.43E+01 ± 1.33E+01 | 1.91E+00 ± 4.33E-01 | 3.68E+00 ± 2.10E+00 | *5.64E-03 ± 1.40E-02* |
| **CPSO-$S_6$** | 10 | 1.41E+00 ± 4.73E-01 | 4.63E-07 ± 6.14E-07 | 1.12E-06 ± 4.01E-07 | 1.39E-01 ± 1.12E-01 | 7.29E-02 ± 1.49E-02 |
| | 15 | 2.47E+00 ± 7.00E-01 | 1.36E-05 ± 1.76E-05 | 1.11E-05 ± 4.53E-06 | 6.00E-02 ± 6.62E-02 | 6.90E-02 ± 1.56E-02 |
| | 20 | 1.59E+00 ± 5.01E-01 | 1.20E-04 ± 8.99E-05 | 5.42E-05 ± 1.66E-05 | 1.46E-01 ± 1.03E-01 | 8.95E-02 ± 1.68E-02 |
| **CPSO-$H_6$** | 10 | 1.94E-01 ± 2.63E-01 | 2.63E-66 ± 5.08E-66 | 9.42E-11 ± 7.58E-11 | 1.47E+00 ± 3.16E-01 | 6.75E-02 ± 1.40E-02 |
| | 15 | 2.59E-01 ± 2.47E-01 | 9.00E-46 ± 1.09E-45 | 9.57E-12 ± 7.96E-12 | 8.77E-01 ± 2.20E-01 | 5.54E-02 ± 1.27E-02 |
| | 20 | 4.21E-01 ± 3.21E-01 | 1.40E-29 ± 1.15E-29 | 2.73E-12 ± 2.03E-12 | 7.78E-01 ± 1.87E-01 | 5.24E-02 ± 1.19E-02 |
| **CPSOLA$_{LRP}$** | 10 | 3.76E-23 ± 8.43E-23 | 5.09E-229 ± 0.00E+00 | 5.42E-14 ± 1.23E-14 | 0.00E+00 ± 0.00E+00 | 3.33E-02 ± 3.83E-02 |
| | 15 | 1.33E-26 ± 4.78E-27 | 3.07E-302 ± 0.00E+00 | 4.99E-14 ± 7.64E-15 | 0.00E+00 ± 0.00E+00 | 2.38E-02 ± 3.07E-02 |
| | 20 | 1.14E-26 ± 3.20E-27 | *3.32e-321 ± 0.00E+00* | 5.28E-14 ± 1.06E-14 | 0.00E+00 ± 0.00E+00 | 4.10E-02 ± 3.95E-02 |
| **CPSOLA$_{RεP}$** | 10 | 1.32E-25 ± 3.02E-27 | 1.72E-306 ± 0.00E+00 | 5.20E-14 ± 1.41E-14 | 0.00E+00 ± 0.00E+00 | 3.85E-02 ± 3.01E-02 |
| | 15 | 1.11E-26 ± 3.14E-27 | 3.61E-311 ± 0.00E+00 | 5.03E-14 ± 1.20E-14 | 0.00E+00 ± 0.00E+00 | 5.39E-02 ± 4.93E-02 |
| | 20 | *1.09E-26 ± 3.36E-27* | 3.75E-321 ± 0.00E+00 | 5.70E-14 ± 1.30E-14 | 0.00E+00 ± 0.00E+00 | 2.50E-02 ± 2.97E-02 |
| **CPSOLA$_{RI}$** | 10 | 1.26E-22 ± 2.45E-22 | 5.09E-229 ± 0.00E+00 | 5.13E-14 ± 9.29E-15 | 0.00E+00 ± 0.00E+00 | 5.80E-02 ± 6.73E-02 |
| | 15 | 1.17E-26 ± 3.59E-27 | 5.25E-301 ± 0.00E+00 | *4.71E-14 ± 1.30E-14* | *0.00E+00 ± 0.00E+00* | 1.53E-02 ± 9.47E-03 |
| | 20 | 1.39E-26 ± 3.96E-27 | 3.93E-321 ± 0.00E+00 | 4.78E-14 ± 8.34E-15 | 0.00E+00 ± 0.00E+00 | 5.53E-02 ± 3.61E-02 |

TABLE II. Learning Automata's Parameters Configuration

| *LA* | *Alpha* | *Beta* | *Action set* | *Initial probability* |
|---|---|---|---|---|
| **$L_{RP}$** | 0.01 | 0.01 | {cooperation, isolation} | {0.5, 0.5} |
| **$L_{RεP}$** | 0.001 | 0.01 | {cooperation, isolation} | {0.5, 0.5} |
| **$L_{RI}$** | 0.01 | 0.00 | {cooperation, isolation} | {0.5, 0.5} |

TABLE III. PSO Algorithm used for Simulation

| **Algorithm** | **Parameters** | **Topology** | **Ref.** |
|---|---|---|---|
| PSO | $w$ = 0.72,$c_1$=$c_2$=2.0 | Global version | [2] |
| CPSO-$S_K$ | $w$:0.9-0.4, $c_1$=$c_2$=1.49,K=6 | Cooperative swarms | [3] |
| CPSO-$H_K$ | $w$:0.9-0.4, $c_1$=$c_2$=1.49,K=6 | Hybrid Co. swarms | [3] |
| CLPSO | $w$:0.9-0.4,c=1.49445,m=7 | Comprehensive learning | [4] |
| CPSOLA | $w$:0.9-0.4, $c_1$=$c_2$=1.49,K=6 | Adaptive H. Co. swarms | [-] |

## B. *Analysis of Results*

Table I shows the averages and standard deviations of the 10 runs of five algorithms on five benchmark functions with 10, 15 and 20 particles. The best results are shown in italic.

*1) Results for the Unimodal Problems:* looking on Table I, we can perceive that in unimodal problems ($f_0$ & $f_1$) CPSOLA's performance is much better than the others. Although standard PSO easily optimized Rosenbrock function; beside their dimension wise search method, CPSO-$S_6$ and CPSO-$H_6$ still trapped in local minima. Thanks to the novel adaptive switching mechanism of learning automaton, CPSOLA algorithm is able to continue improving its performance while the others trapped in pseudominima. The Quadric function presents similar results; CPSOLA performs significantly better than the other PSOs. Since CLPSO has a wide search space, it could not converge as quickly as the other algorithms and get the worst results.

*2) Results for the Multimodal Problems:* there are three test functions which placed in this group ($f_2$-$f_4$). As can be seen in Table I, standard PSO and CLPSO algorithms are trapped in a local minimum; while CPSO-$S_6$ and CPSO-$H_6$ are achieve better solutions. The best result belongs to CPSOLA$_{RI}$ algorithm. Scheme of Ratrigin's function is similar to Ackley, which trigger a same performance between PSOs. In CPSOLA, all three types of learning automatons achieve the best results. Also CPSO-S and CPSO-H are optimized this problem as well as CPSOLA algorithms [3]. Since in CLPSO each dimension of particles' exemplar select through a tournament selection, CLPSO achieved the best results on Griewank's function. The results of CPSOLA are as well as CPSO-$S_6$ and CPSO-$H_6$.

*3) Population Size:* The experiments which are conducted on 30–D problems are repeated with 10, 15 and 20 particles. In Table I, the S entry indicates the population size. The results show, increasing the number of particles and keeping the problem's dimension fixed, will lead to improve the fitness value. The CPSOLA performs better when using 20 particles per population.

*4) Convergence Graph of Ackley's Function:* the Ackley's function has an exponential term which caused its surface to cover with several local minima [3]. Fig. 6 shows the convergence characteristics of 30– dimensional Ackley' function with 20 particles. The first flat line in Fig. 6 indicates

that the Standard PSO becomes trapped in a local minimum. Since CLPSO has a large feasible search space, it is trapped in a local minimum either; the second flat line shows that. From the third and fourth flat lines which belong to CPSO-$S_6$ and CPSO-$H_6$ algorithms, we can observe that they have a fast convergence speed and these results are Due to the exhaustive dimension wise search method of Cooperative PSO. The adaptive switching mechanism of CPSOLA has a cost and the cost is the slow convergence of the algorithm. Since using the alternative population may suppress improving the global best particle of primary population for a while, CPSOLAs are managed to continue improving their performance very well. The best result belongs to $CPSOLA_{RI}$, which its learning algorithm can find the optimal policy faster than the others. The $L_{RI}$'s early decision making property helps $CPSOLA_{RI}$ to escape form the local minima before it's too late.

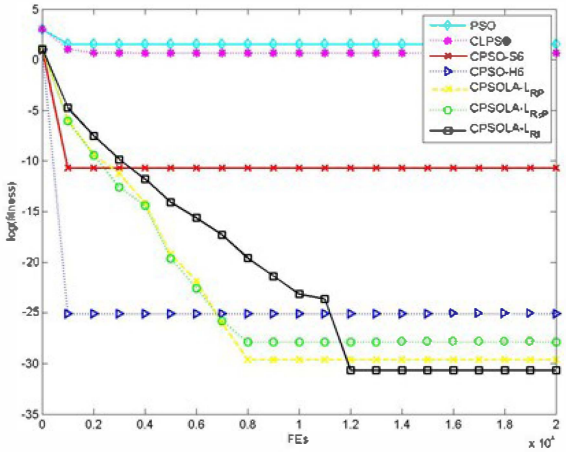


Figure 6. Ackly's function ($f_2$) convergence graph.

*5) Automaton's Learning Algorithm Comparison:* Since the *learning parameters* are set small for $L_{RP}$ and $L_{R\varepsilon P}$ learning algorithms, they perform better when the problem is unimodal and the automaton should learn in a simple search space. Having zero penalty value, the $L_{RI}$ learning algorithm could choose optimal action before get trapped in a local minimum. The $L_{RI}$ learning algorithm is suitable for multimodal problems.

*6) Robustness*: a "Robust" algorithm is one that can decrease the fitness below a specified threshold in a fewer number of fitness evaluations [3]. Although no specific robustness experiment is reported, Table I shows that all CPSOLA algorithms solve the problem with low variances. This indicates that the CPSOLA algorithm is consistently managed to optimize the problems.

## VI. CONCLUSION

In this paper we presented a cooperative particle swarm optimizer with adaptive control on the outer layer of cooperation. The results are shown a significant improvement in performance and robustness. Like CPSO-$H_K$ algorithm in CPSOLA we have two populations: The first one named as primary population and belong to CPSO, while the secondary one belongs to PSO algorithm. In the proposed approach a learning automaton observe the global best fitness of primary population and decide when to cooperate with secondary one. Having a comprehensive scheme of problem to be optimised, the learning automaton controls the evolution process. Since the evolution of secondary population may lag from the first one, the automaton brings an indirect diversity while switching between its actions. In the real world every action has a consequence; slow convergence is the cost that we paid for our algorithm. The CPSOLA performs the best in unimodal test functions and the $L_{RI}$ learning technique is a better choice in multimodal problems. Totally the CPSOLA is shown a better and reliable performance in four out of five test functions.

## ACKNOWLEDGMENT

This work is supported by Iran Telecommunication Research Center (ITRC) grant.

## REFERENCES

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995,* pp. 39-43.

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

[3] 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, Jun. 2004.

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

[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, Jun. 2011.

[6] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

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

[8] C. Ünsal, "Intelligent navigation of autonomous vehicles in an automated highway system: Learning methods and interacting vehicles approach," Virginia Polytechnic Institute and State University, 1997.

[9] 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, Jan. 2011.

[10] A. Rezvanian and M. R. Meybodi, "LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization," in *3rd International Conference on Contemporary Computing,* 2010, vol. 94, pp. 64–75.

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

[12] M. Hamidi and M. R. Meybodi, "New Learning Automata based Particle Swarm Optimization Algorithms," presented at the Iran Data Mining Conference, 2008, pp. 1-15.

[13] M. Hasanzadeh, M. R. Meybodi, and S. Shiry, "Improving Learning Automata based Particle Swarm: An Optimization Algorithm," in *12th IEEE International Symposium on Computational Intelligence and Informatics*, 2011.

[14] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature*, pp. 249–257, 1994.

[15] F. van den Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, pp. 84–90, 2000.