A Learning Automata Approach to Cooperative Particle Swarm Optimizer

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

Received: 14/Apr/2013 Accepted: 08/Feb/2014

Abstract
This paper presents a modification of Particle Swarm Optimization (PSO) technique based on cooperative behavior of swarms and learning ability of an automaton. The approach is called Cooperative Particle Swarm Optimization based on Learning Automata (CPSOLA). The CPSOLA algorithm utilizes three layers of cooperation which are 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 consists of multiple dimensions of search space. Also there is a secondary population in CPSOLA which is used the conventional PSO's evolution schema. In the upper layer of cooperation, the embedded Learning Automaton (LA) is responsible for deciding whether to cooperate between these two 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 Optimizer (PSO), Cooperative Particle Swarm Optimizer (CPSO), Learning Automata.

1. 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], [4] 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 particles, the cooperation is occurred between swarms. Comprehensive Learning PSO (CLPSO) [5] 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 [6] 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) [7] 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 by four strategies named as exploration, exploitation, convergence and jumping out.

A Learning Automaton (LA) [8], [9] 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) [10], [11]. Also a new hybrid method of optimization which called PSO-LA [12]–[17] has been emerged. In PSO-LA algorithms an LA or a group of learning automata 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. Moreover, LA has application in Grid computing [18]. In [19], Distributed Learning Automata (DLA) has been used for Grid resource discovery.

CPSO family [3] consists of four algorithms: CPSO-S, CPSO-S_K, CPSO-H and CPSO-H_K where 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 emerged as the 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 [16] the first attempt to improve this hybridization of PSO and CPSO algorithms is done by embedding an automaton as a toolbox of the switching mechanism. Furthermore, in this paper we deeply investigate the behavior of discussed learning automata approach for CPSO family by a set of diverse experiments.

* Corresponding Author

The paper is organized as follow: section 2 reviews two PSO heuristics. 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.

2. Particle Swarm Optimizer (PSO)

2.1 Conventional formulation of PSO

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 are initialized in a feasible search space. The velocity, position and best previous position of i_{th} particle are respectively shown by $X_i = (x_i^1, x_i^2, \dots, x_i^N)$, $V_i = (v_i^1, v_i^2, \dots, v_i^N)$ and $pbest_i = (pbest_i^1, pbest_i^2, \dots, pbest_i^N)$. Also the best position of the population is $gbest = (gbest^1, gbest^2, \dots, gbest^N)$. 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 [5]:

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

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

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 searches. *rand1* and $rand2 \in [0,1]$ are two random numbers generated for each dimension of the particles. The algorithm of the original PSO is given in Fig. 1.

2.2 Cooperative Learning in PSO

The idea of cooperative learning was first implemented in the field of Genetic Algorithm (GA) by Potter [20]. 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 cooperative technique in PSO may lead to promising results. Recently The concept of cooperation mapped into PSO technique. Cooperative behavior in PSO was first introduced by Van den Bergh [4]. 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 swarms are trying to find a better solution vector. 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* swarms 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 1st phase CPSO-S_K run and the information exchange performs *from* CPSO-S_K half *to* PSO half of algorithm. At 2nd phase, PSO run and information exchange *form* PSO half *to* CPSO-S_K half performs. Note that each phase performs in a separate iteration.

Cooperative PSO [3], [4] divides the initial population into some subpopulations and each of these subswarms optimizes their designated dimensions individually. There are two layers of cooperation in a cooperative PSO. The first layer lies under the collaborative behavior of particles in specific dimensions and the second one is the schema that produces a solution vector by means of sharing the best information of each subpopulation to constitute a valid solution vector. In order to evaluate each member of the subpopulation, one requires constructing a *context vector* which aggregates the best solution of each subpopulation within an *N*-dimensional vector. Typically to evaluate the current subpopulation, the corresponding dimensions filled with the position of particle and the other dimensions are considered constant. Fig. 2 is the cooperative PSO pseudocode.

Algorithm 1 Standard PSO

for each generation *k* **do**
  for each particle *i* **do**
    Update velocity of i_{th} particle by (1)
    Update position of i_{th} particle by (2)
    Calculate particle fitness $f(x_i)$
    Update $pbest_i$ and *gbest*
    i = *i*+1 // next particle
  end for
  k = *k*+1 // next generation
end for

Fig 1. The pseudocode of the standard PSO

Algorithm 2 Cooperative PSO (CPSO)

define
Split *N*-dimensional search space into *j* subpopulations.
Calculate the best individual of each subpopulation (*sbest*).
Construct a *Context Vector* **(CV)** through the best individuals of each subpopulation:
CV = [$sbest_1$, $sbest_2$, …, $sbest_j$]
for each generation *k* **do**
  for each subpopulation *j* **do**
    for each particle *i* **do**
      Replace current particle of j^{th} Subpopulation by its

```
            corresponding positions in the CV
            Evaluate the N-dimensional output vector through
            the fitness function.
            i = i+1 // next particle
        end for
        Update sbest_j.
        j = j+1 // next swarm
    end for
    k = k+1 // next generation
end for
```

Fig. 2 The pseudocode of the cooperative PSO

2.3 Cooperative based PSO algorithms

As well as cooperative PSO [21] and GA [22], the cooperative approach is also implemented in other EAs such as: Evolutionary Strategy (ES) [23], Differential Evolution (DE) [24], [25] and Artificial Bee Colony (ABC) [26]. The following are four recent advances in the context of cooperative PSO:

The Cooperative Coevolutionary ES (CCES) [23] divides the population of ES into some subspecies and lets them evolve. By means of a migration operator, CCES could hybridize the cooperative evolutionary behavior of CPSO with ES. The proposed model controls the interaction of subspecies properly and exhibits good performance results.

The Cooperative Coevolutionary DE (CCDE) [24] partitioned the problem into several sub problems and allocated a subpopulation to each of them. In [25], a randomized grouping mechanism introduced and an adaptive weighting strategy used in order to adapt the separated components. The idea was accomplished to bring the interacted variables into a similar subcomponent.

The self-adaptive neighborhood search into DE (SaNSDE) could tackle the non-separable problems with more than 1000 dimensions inside.

The cooperative approach of Potter is exerted into ABC and Cooperative ABC (CABC) [26] is emerged. Like two variants of CPSO, he introduced two versions of split swarm and hybrid for CABC. The CABC_S algorithm can efficiently optimize the separable problems and the CABC_H algorithm has the ability to escape from the local minima.

2.4 Evaluation Scheme of PSO versus CPSO

The key characteristic of Standard PSO [2] and cooperative PSO [3] is related to their corresponding population. The standard PSO contains a single population where this single population is divided into multiple swarms in cooperative PSO.

There is a paradigm in conventional PSO algorithm which could be extended to Cooperative PSO: *In order to find a proper solution vector, each particle of the swarm fly through an N–dimensional search space by N values corresponded to each dimension of the space*. To understand this phrase deeply, consider the population as a matrix $_{[M\times N]}$ where M and N respectively represent the number of particles and dimensions, respectively as: $[\#of\ Particles \times \#of\ Dimensions]$ (see Fig. 3). In this framework, velocity and position of the standard PSO [2] population were updated row wise. The interpretation of this framework in CPSO [3] is quite different from that of standard PSO. In CPSO the population is optimized column wise (dimension wise) with the dimension of each particle being evaluated by a *context vector (CV)* which is built from the best particle of corresponding swarm and the best particles of other swarms.

$$\begin{bmatrix} P/D & \boxed{D_1} & \boxed{D_2} & \cdots & \boxed{D_N} \\ \boxed{P_1} & P[1,1] & P[1,2] & \cdots & P[1,N] \\ \boxed{P_2} & P[2,1] & P[2,2] & \cdots & P[2,N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boxed{P_M} & P[M,1] & P[M,2] & \cdots & P[M,N] \end{bmatrix}$$

$$PSO: \quad f(P_i) = fitness(P_i(1 \to D))$$

$$CPSO: \quad f(P_i, S_j) = fitness(CV(P_i, S_j, j))$$

Fig. 3 Comprehensive view of the PSO population. $f(P_i)$ represents the i_{th} particle of population which evaluates through the traditional PSO mechanism and $f(P_i,S_j)$ indicates the evaluation process of the i_{th} particle (P_i) of j_{th} swarm (S_j) of CPSO population.

3. Learning Automata (LA)

3.1 Conventional Formulation of LA

Learning Automata [8], [9] 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. 4 shows how automaton interacts with its environment. A study of the learning process of LA in a random environment is comprehensively reported in [8].

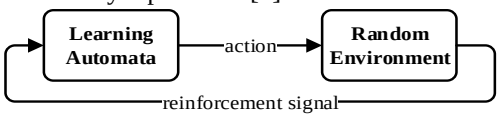


Fig. 4. The interaction between learning automata and environment.

Learning automata [8], [9] are divided into two groups of *fixed-structure* and *variable-structure automata*. A Variable-Structure LA (VSLA) is represented by a quadruple $[\alpha, \beta, p, T]$, Where $\alpha = \{\alpha_1, \dots, \alpha_r\}$ is a set of actions, $\beta = \{\beta_1, \dots, \beta_r\}$ is a set of inputs, $p=\{p_1,\dots,p_r\}$ 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, the action probabilities are updated through (3):

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

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

3.2 LA based PSO algorithms

Parameter adaption [10], [11] is one of the most difficult tasks in EAs. As there are multiple parameters in PSO, it needs a mechanism to tune them during the evaluation of the population. In [10] a study of adaptive PSO parameter selection is conducted. Also, 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 (|LA|=1). Since coarse-grained PSO-LA algorithms are trapped into local minima, in fine-grained PSO-LA algorithms [13], [14], learning automata are assigned to each particle of the swarm (|LA|=|*population size*|). This technique gives more maneuverability to the particles for searching through the search space. Some improved PSOs using LA are illustrated in the following:

A Dynamic Particle Swarm Optimization based on a 3-action Learning Automata (DPSOLA) introduced in [14]. 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.

One variant of PSO is Comprehensive Learning Particle Swarm Optimizer (CLPSO) [5], which uses all individuals' best information to update their velocity. The novel strategy of CLPSO enables population to read from exemplars for specified generations which is called refreshing gap m. In [15], two classes of Learning Automata (LA) developed in order to study the learning ability of automata for CLPSO refreshing gap tuning. In the first class, a learning automaton is assigned to the population and in the second one each particle has its own personal automaton.

The cooperative PSO based on LA which introduced in [16] is the first version of CPSOLA. This algorithm utilizes both beneficial characteristics of PSO and CPSO by employing a learning automaton as a realtime decision making optimization tool.

The Adaptive Cooperative Particle Swarm Optimizer (ACPSO) [17] which facilitates cooperation technique through usage of LA algorithm. Cooperative learning strategy of ACPSO optimizes the problem collaboratively and evaluates it in different contexts. In ACPSO algorithm, a set of learning automata associated with dimensions of the problem are trying to find the correlated variables of the search space and optimize the problem intelligently. This collective behavior of ACPSO will fulfill the task of adaptive selection of swarm members.

4. Cooperative Particle Swarm Optimizer based on Learning Automata (CPSOLA)

4.1 Presenting Non-constraint Optimization Problems

Optimization is an approach to solve the complicated problems, like scheduling tasks. The framework of an *N*-dimensional, non-constraint optimization problem is as follows [5]:

$$\min\left\{ f(x) \right\}; \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad x_1 \le x \le x_n \quad (5)$$

The optimization aim is to seek the optima in a search space by generating several feasible solutions and selecting the optimum one. Usually traditional techniques like dynamic programming and greedy algorithms are applied to optimization problems. In dynamic programming by using divide and conquer method, in each interval a portion of the search space will be omitted and the optimal solution will extract from the residual information. Also, during some steps of greedy algorithms, the most optimal solution will be selected by an excessive policy. In both of these listed algorithms, the election of each step is performed the local search with the hope of discovering the best solution. The lake of retaining the exploration and exploitation equilibrium and also excreting a segment of the search space in each epoch, are the defects of the conventional optimization approaches.

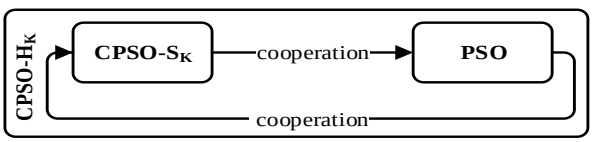


Fig. 5. Structural view of the CPSO-H_K algorithm.

4.2 Introducing the Non-adaptive Cooperative 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. 5 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:

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.

4.3 Describing the Adaptive Cooperative Behavior of CPSOLA

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 CPSOLA algorithm presents two advantages in contrast to CPSO-H_k:

1) CPSOLA simultaneously utilizes both beneficial characteristics of PSO and CPSO-S_k. It inherits fast convergence speed of CPSO and also easily escapes from local minima by utilizing PSO's *gbest* and *pbest* information.

2) CPSOLA adaptively uses both CPSO-S_K and PSO in order to maintain marginal performance while CPSO-H_K interleave this algorithms without any environment perception.

3) CPSOLA balance the global and local search by maintaining population diversity between PSO and CPSO.

The scheme of adaptive switching mechanism of CPSOLA is shown in Fig. 6. 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.

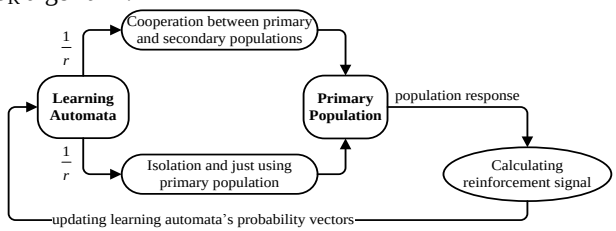


Fig. 6. Schematic view of decision making by learning automata

In each generation of CPSOLA algorithm, the automaton decides whether to start the alternative population (PSO population) or not. While having enough interactions with the populations, the automaton perceives when to switch between two algorithms. In the following we listed the five main differences between the CPSOLA model 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 population.

Equation (6) is the criterion to evaluate the reinforcement signal in CPSOLA algorithm. *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.

$$\beta = \begin{cases} 0 & if \quad fitness\left(Sbest_i < Sbest_{i-1}\right) \\ 1 & Otherwise \end{cases} \tag{6}$$

Fig. 7 is the pseudocode of CPSOLA algorithm. LA has two actions: first, cooperation or information exchange between CPSO and PSO population and second, isolation or evolution of CPSO population. In CPSOLA algorithm, an LA determines the time to perform cooperation between primary and secondary populations. At the earliest generations the LA selects actions randomly but after passing middle iterations the ultimate perception of the optimization problem occurs and the LA could accurately detects the time for switching between these two populations. This switching time is a key advantage of CPSOLA algorithm in contrast to the CPSO-H_K interleave switching strategy.

Algorithm 3 Cooperative PSO based on LA (CPSOLA)

```
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 ∈ [1..K]
        for each particle i ∈ [1..s]
        Update particle position by Equations (1) & (2)
            Calculate particle fitness
            Update pbest & gbest
            i = i + 1 // next particle
         end for
         j = j + 1 // next swarm
      end for
      Select a random particle form Q to write
      for each particle Q; i ∈ [1..s]
        Update particle position by Equations (1) & (2)
         Calculate particle fitness
         Update pbest & gbest
         i = i + 1 // next particle
      end for
      for each swarm P_j; j ∈ [1..K]
         Select a random particle form P to write
      end for
   else if the selected action is isolation then
      for each swarm P_j; j ∈ [1..K]
        for each particle i ∈ [1..s]
         Update particle position by Equations (1) & (2)
            Calculate particle fitness
            Update pbest & gbest
            i = i + 1 // next particle
         end for
         j = j + 1 // next swarm
      end for
   Evaluate reinforcement signal by Equation (6)
   Update LA's probability vectors by Equations (3) & (4)
k = k+1 // next generation
until a terminate condition is met
```

Fig. 7. The pseudocode of the CPSOLA algorithm.

4.4 Analyzing the Adaptive Cooperative Behavior of CPSOLA

Action selection of LA needed 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 process. The fitness comparison of CPSO-S_K and PSO are plotted in Fig. 8.a. By looking on zoomed boxes of specific parts of evolution, we can observe that, in the earliest iterations of CPSOLA algorithm, there is no obvious difference between the fitness of two populations. The algorithm probably could escape from the local minima after reaching the middle iterations, so the policy is changed adaptively and even during this part of evolution the PSO's fitness even could be better than CPSO-S_K's fitness. Because of its cooperative search method of CPSOLA, in the last generations, the CPSO-S_K's fitness becomes dominated and the algorithm converges faster than original CPSO algorithm.

The variance of action probabilities are plotted in Fig. 8.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 performs during 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 easily could escape from the local minimum.

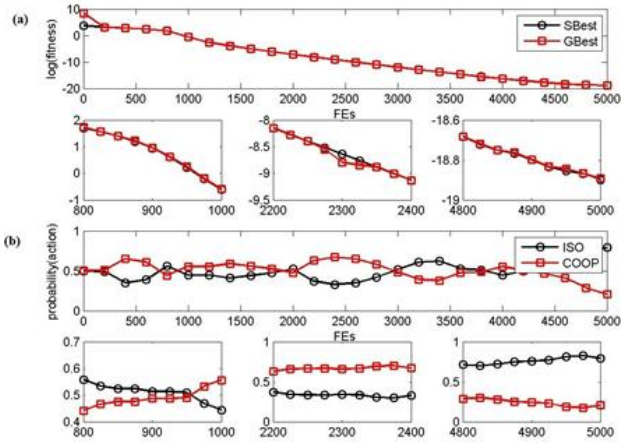


Fig. 8. (a) The CPSO-S_K's Swarm Best Position (*sbest*) versus the PSO's global best position (*gbest*). The horizontal and vertical axis of each box represents the number of fitness evaluations (FEs) and the fitness value in logarithmic scale, respectively. (b) Variance of LA's actions probability. The horizontal and vertical axis of each box represents the number of fitness evaluations (FEs) and the probability value in Gaussian scale, respectively.

5. Experimental Study

5.1 Simulation Setup

In order to have a fair comparison, all the PSOs should use a same number of fitness. This number is set to 20000. 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.

Observing the learning automaton's [10], [11] behavior during the evolution process, three different kind of learning algorithms are placed in CPSOLA algorithm [16]. The detailed configurations of the reward and penalty parameters are mentioned in Table I. As long as the L_{RP} learning algorithm acts as a moderate one among three of them, in the following we just report the results of this learning algorithm.

In order to compare the proposed method, we simulate four other PSOs including: standard PSO algorithm [2], Split swarm Cooperative PSO (CPSO-S) [3], Hybrid Cooperative PSO (CPSO-H) [3] and Comprehensive Learning PSO (CLPSO) [5]. The settings of these PSOs are briefly listed in Table II.

We choose five benchmark functions [3], [5] to run the experimental tests. Among them, f_0 (Rosenbrock) and f_1 (Quadric) functions are simple unimodal problems which have non-complex structure. The functions f_2 (Ackley), f_3 (Rastrigin) and f_4 (Griewank) functions are highly multimodal problems with many local optima positioned in their grid. The details and mathematical formulation of these benchmark functions are depicted in Table III.

Unimodal functions have simple structure and small number of minima. These kinds of optimization benchmarks can be easily optimized by deterministic optimization algorithms which use gradient information of benchmark. The first problem, function f_0 is a non-convex function that its global optima is placed in a long, narrow and hyperbolic valley. This function can be treated as a multimodal function. The function f_1 is a simple function that usually used for showing the power of optimization algorithms in function optimization. It is a separable benchmark function that each dimension could optimize independently.

Table I: The parameters configuration of learning automata. The **LA** column indicates the learning algorithm. The **Alpha** and **Beta** columns indicate different values of reward and penalty signals for each learning algorithm. Also the **Action set** column shows two actions (*cooperation and isolation*) of learning automata and the **Initial probability** of 0.5 indicates the raw probability of each action

LA	Alpha	Beta	Action set	Initial probability
L_{RP}	0.01	0.01	{cooperation, isolation}	{0.5, 0.5}
$L_{R\varepsilon P}$	0.001	0.01	{cooperation, isolation}	{0.5, 0.5}
L_{RI}	0.01	0.00	{cooperation, isolation}	{0.5, 0.5}

Table II: The PSO algorithms used for simulation. Each algorithm has unique parameter settings which are derived from its associated reference. **The Algorithm** column lists the different PSOs that used for simulation. The **Parameters** column includes the parameter used for each PSO. The **Topology** column shows the special attribute of each PSO and the **Ref**. column represents the associated reference number

Algorithm	Parameters	Topology	Ref.
PSO	$w = 0.72, c_1=c_2=2.0$	Global version	[2]
CPSO-S_K	$w:0.9\text{-}0.4, c_1=c_2=1.49, K=6$	Cooperative swarms	[3]
CPSO-H_K	$w:0.9\text{-}0.4, c1=c2=1.49, K=6$	Hybrid Cooperative swarms	[3]
CLPSO	$w:0.9\text{-}0.4, c=1.49445, m=7$	Comprehensive Learning Strategy	[4]
CPSOLA	$w:0.9\text{-}0.4, c_1=c_2=1.49, K=6$	Adaptive Hybrid Cooperative swarms	[-]

Most of optimization heuristics suffer from the *curse of dimensionality* [27]. This phenomenon appears when the performance of algorithm degraded rapidly by increasing the dimensionality of the problem. This situation has two reasons: 1) By increasing the dimensionality of benchmark functions, the number of local minima increases exponentially. A successful algorithm in this situation is one that can search more promising regions of search space. 2) Some benchmark functions are reshaped by increasing the number of dimensions. For example, function f_2 is a unimodal function in low number of dimensions (2 dimensions). Also, in this function by increasing the number of dimensions it will convert to a multimodal function. Due to the aforementioned reasons of curse of dimensionality, some search strategies which may work well in low dimensional benchmark functions can't find the optimum solution in high dimensional benchmark functions.

Multimodal benchmark function (f_2-f_4) has many local optima and one global optimum. These benchmark functions have complicated structure. The convergence speed of multimodal functions is lower than unimodal functions.

The function f_2 is a separable function with an exponential term which covers the surface of function with many local minima. It has one narrow global optimum basin and many minor local optima. Simple gradient descent algorithms are failed to optimize this function, but any heuristic which can move through the valley of the function can attain better results.

The function f_3 is a complex multimodal problem with large number of local optima. When attempting to solve function f_3, algorithms may easily fall into a local optimum. Hence, an algorithm capable of maintaining larger population diversity is likely to yield better results.

The function f_4 has a $\prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right)$ component causing linkages among variables, thereby making it difficult to reach the global optimum. An interesting phenomenon of function f_4 is that it is more difficult for lower dimensions than higher dimensions.

Table III: The formulation of benchmark functions. The **Function** column shows five different benchmark functions. The **Formula** column shows the mathematical formulation of each benchmark function. The values listed in the **Range** column are used to specify the magnitude to which the initial random particles are scaled. The V_{max} column indicates the maximum velocity which is used to clamp the velocity of particles. The **Threshold** column lists the function value threshold which is used as a stopping criterion in section 5-4. The **Shape** column shows a 2D view of each benchmark function

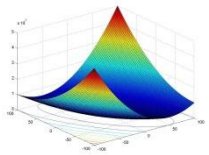
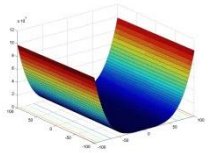
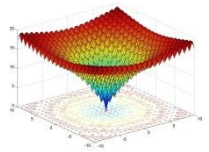
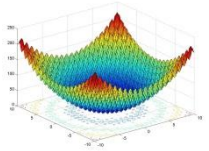
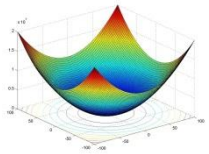
Function	Formula	Range	V_{max}	Threshold	Shape
f_0 (Rosenbrock)	$f_0(x)=\sum_{i=1}^{\frac{n}{2}}\left(100\left(x_{2i}-x_{2i-1}^2\right)^2+\left(1-x_{2i-1}\right)^2\right)$	[-2.048, 2.048]	2.048	0.01	
f_1 (Quadric)	$f_1(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	[-100, 100]	100	0.01	
f_2 (Ackley)	$f_2(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	[-32, 32]	32	0.01	
f_3 (Rastrigin)	$f_3(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos\left(2\pi x_i\right)+10\right)$	[-5.12, 5.12]	5.12	0.00001	
f_4 (Griewank)	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]	600	0.01	

Table IV: The average and standard deviation of PSO, CLPSO, CPSO-S_6, CPSO-H_6, $CPSOL_{RP}$, $CPSOL_{R\varepsilon P}$ and $CPSOL_{RI}$ algorithms over 10 independent runs on 5 benchmark 30 – dimensional functions with 20000 Fitness Evaluations. The second column **S** lists the number of particles per swarm

Function / Algorithm	S	f_0 (Rosenbrock)	f_1 (Quadric)	f_2 (Ackley)	f_3 (Rastrigin)	f_4 (Griewank)
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	0.00E+00 ± 0.00E+00	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	0.00E+00 ± 0.00E+00	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	0.00E+00 ± 0.00E+00	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	0.00E+00 ± 0.00E+00	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	0.00E+00 ± 0.00E+00	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	0.00E+00 ± 0.00E+00	5.24E-02 ± 1.19E-02
$CPSOL_{RP}$	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

5.2 Experiment 1: Function Optimization

Table IV shows the mean and standard deviation of 10 runs of each PSO on 30 dimensional problems. In unimodal functions cooperative PSOs could reach good results. The function f_0 is a simple unimodal function and its global optima places in the center of function. Also, function f_1 is a unimodal function with one global optimum. Standard PSO performs better in f_1 when comparing with CPSO family and CLPSO. This implies that the CPSO is less effective in solving simple problems. In the other hand, due to adaptive switching mechanism of CPOSLA, the algorithm could use the PSO's dimension-wise updating rule while using several swarms to optimize different dimensions of search space independently. By combining the results of these two properties, CPSOLA achieves the best results on unimodal functions (f_0 and f_1).

In Table IV, functions f_2-f_4 are multimodal functions. In f_2 and f_3 CPSOLA and CPSO generally outperform standard PSO, CLPSO that involves neither cooperative swarms nor information exchange property. However, the CPSOLA is most powerful and robust for these two test problems. These results confirm the hypothesis that cooperative swarms speed up the convergence of CPSOLA algorithm and adaptive switching mechanism helps the swarms jump out of the local optima and find better solutions.

Different dimensions of CLPSO may learn from different exemplars. Due to this, the CLPSO explores a larger search space than the other PSOs in function f_4. The larger search space is not achieved randomly. Instead, it is based on the historical search experience. Because of this, the CLPSO performs comparably to or better than other PSO variants on function f_4. Note that function f_4 is known to become easier as the number of dimensions increases.

The experiments which are conducted on 30–D problems are repeated with 10, 15 and 20 particles. In Table IV, 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 of PSOs. Although from the results of Table IV the resulting performance may vary depending on the problem being optimized. There seems to be no definitive value for the swarm size that is optimal across all problems, so to avoid tuning the algorithm to each specific problem, a compromise must be reached. 15 particles were shown the best results for CPSOLA, as populations of this size performed best by a very slight margin when averaged across the entire range of test problems.

5.3 Experiment 2: Convergence graph

Fig. 9 presents the convergence characteristics in terms of the mean best fitness value of each algorithm for all 30 – dimensional test function with 20 particles. The comparisons in both Table IV and Fig. 9 show that, when solving unimodal and multimodal problems, the CPSOLA offers the best performance on most test functions. In particular, the CPSOLA offers the highest accuracy on functions f_0, f_1, f_2 and f_3. Furthermore, CLPSO shows the best convergence characteristics on function f_4.

(a)

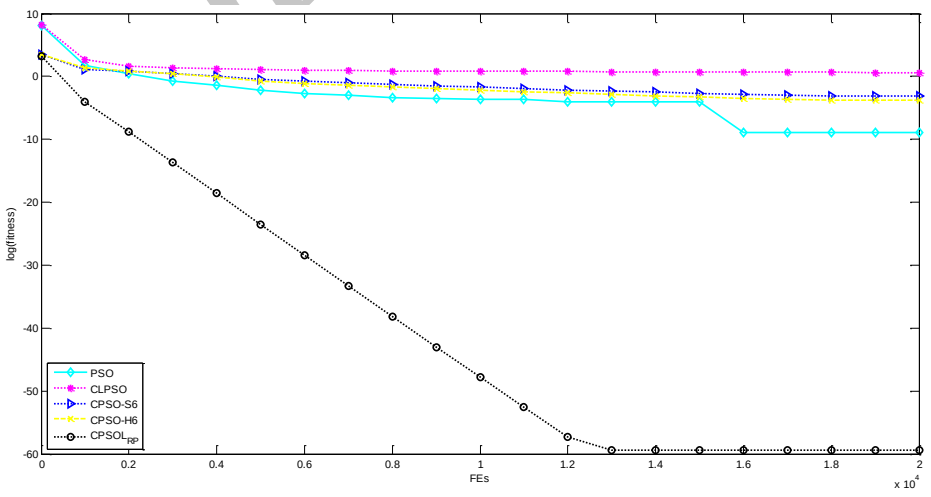


Fig. 9. The mean convergence graph of 30-D benchmark functions. (a) f_0 (Rosenbrock).

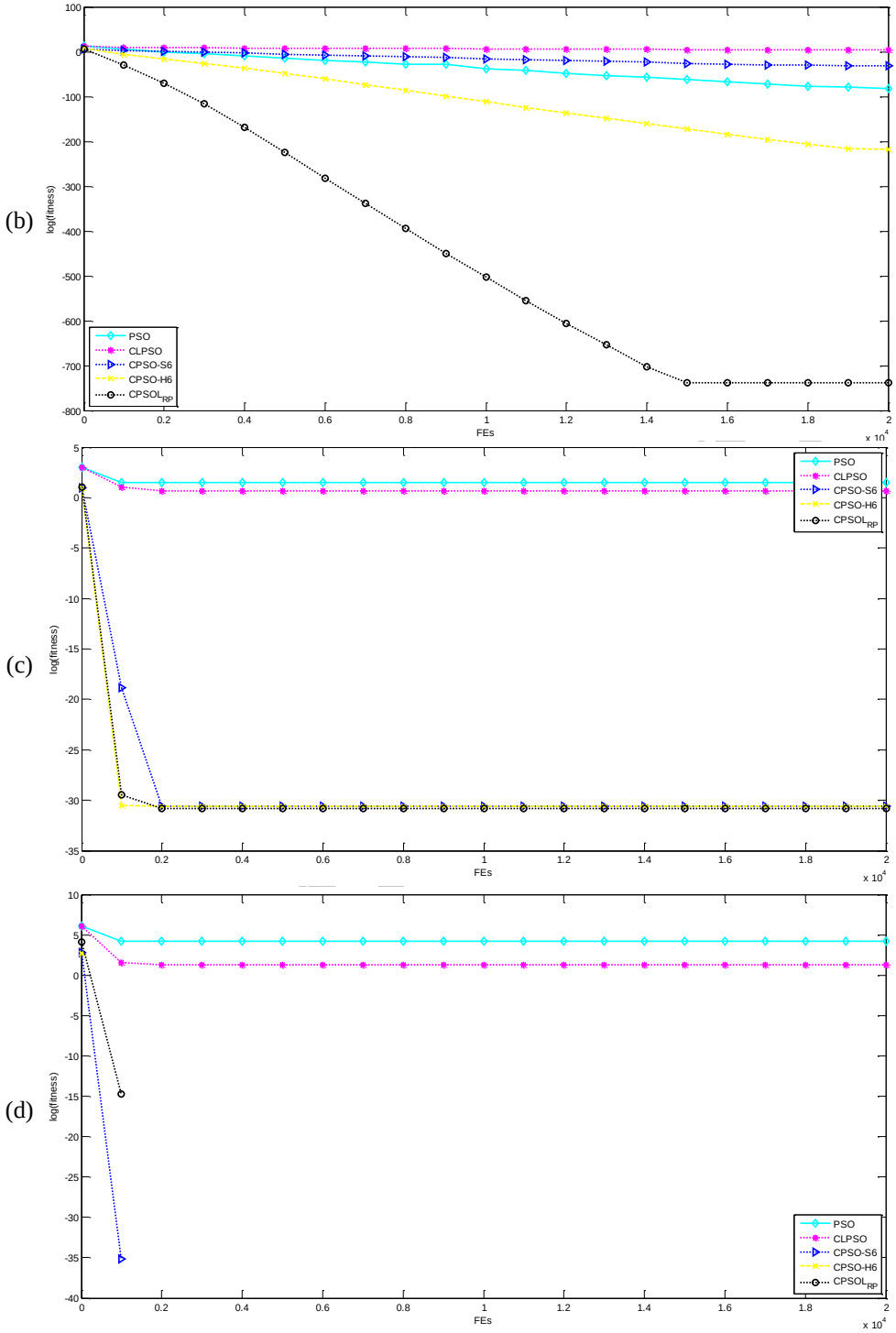


Fig. 9. (Continued.) The mean convergence graph of 30-D benchmark functions. (b) f_1 (Quadric). (c) f_2 (Ackley). (d) f_3 (Rastrigin).

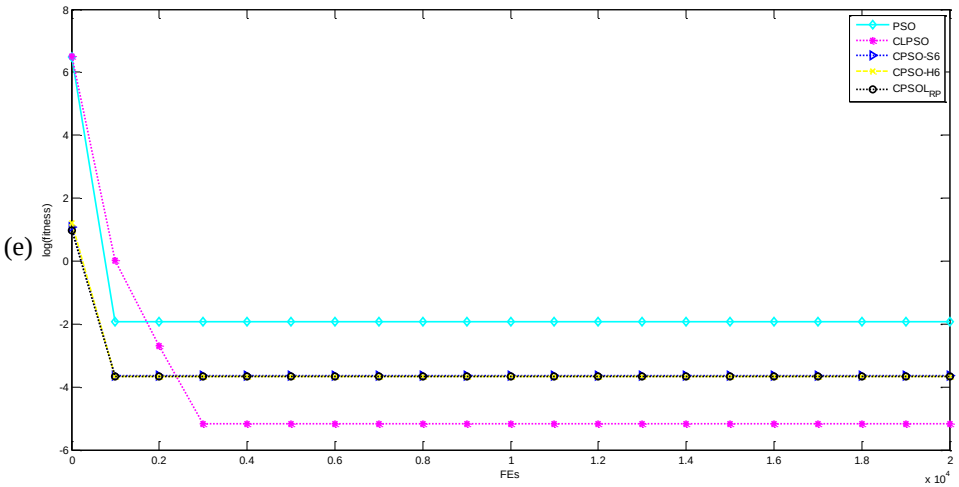


Fig. 9. (Continued.) The mean convergence graph of 30-D benchmark functions (e) f_4 (Griewank)

The Rosenbrock function (f_0) is a non-convex test problem with a long, narrow and parabolic shaped flat valley. From the set of flat lines of Fig. 9 – a (f_0), CLPSO and CPSO are easily get trapped in local minima while standard PSO still improves its fitness value. Moreover, CPSOLA could escape from the local minima till the middle generations and acquire the best fitness value. CPSOLA can find the best trail into the valley and almost converges to the global minimum.

The Quadric function (f_1) is a convex and unimodal problem which has D local minima except for the global one. In Fig. 9 – b (f_1) CPSOLA and CPSO-H_6 converge better than other PSOs. Both of these cooperative heuristics utilize two populations in order to optimize the test problem. Since CPSOLA uses adaptive switching mechanism while CPSO-H_6 uses interleave switching mechanism, the CPSOLA approach could exploit beneficial properties of PSO and CPSO-S_k algorithms.

The Ackley function (f_2) is a non-separable and multimodal function. It has a flat outer region and a large hole at the center. The function poses a risk for optimization heuristics, particularly hill climbing and gradient steepest descent algorithms. Fig. 9 – c (f_2) shows the convergence characteristics of 30– dimensional Ackley' function with 20 particles. The first flat line in Fig. 9 – c indicates that the Standard PSO becomes trapped in a local minimum in early generations. Since CLPSO has a large feasible search space, it is easily 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 approach. The adaptive switching mechanism of CPSOLA has a cost and the cost is the slow convergence of the algorithm in this problem. Since using the alternative population may suppress improving the global best particle of primary population for a while, CPSOLA is managed to continue improving its performance very well. The best result belongs to $CPSOL_{RP}$, which its learning algorithm can find the optimal policy faster than the others. Also definitive decision making property of LRP learning algorithm helps $CPSOL_{RP}$ algorithm to escape form the local minima before it becomes too late.

The Rastrigin function (f_3) is a non-convex, non-linear and multimodal optimization problem that has large number of local minima whose values increases with the distance to the global minimum. Finding the minimum of Ackley function is fairly difficult problem due to Ackley's large search space and its large number of local minima. From Fig. 9 – d (f_3), PSO and CLPSO are get trapped in local minima in early generations but all cooperative PSOs converge to global optimum. Also the CPSOLA algorithm can converge faster than CPSO-S_6.

The Griewank function (f_4) is a multimodal and non-separable test function. The algorithms which try to optimize each variable of this benchmark function independently, lead to failure. The global optimum of this function is regularly distributed. From the results presented in Fig. 10 – e (f_4), it can be observed that all PSO variants failed on Griewank function except for CLPSO. The algorithm utilizes a tournament selection heuristic for determining the exemplar in each dimension.

Figures 9 – a to e show the plot performance of the PSO, CLPSO, CPSO-S_6, CPSO-H_6 and $CPSOL_{RP}$ algorithms in 30-dimensional problems. The CPSOLA algorithm uses a bi-action learning automaton with the L_{R-P} learning algorithm.

Table V: Robustness Analysis. The second column **S** lists the number of particles per swarm. The column **Succeeded** list the number of runs (out of 10) that manage to attain a function value below the threshold in less than 20000 fitness evaluations, while the column **FEs** presents the number of function evaluations needed on average to reach the threshold, calculated only for runs that **Succeeded**

Function / Algorithm	S	f_0 (Rosenbrock)		f_1 (Quadric)		f_2 (Ackley)		f_3 (Rastrigin)		f_4 (Griewank)	
		Succeeded	**FEs.**	**Succeeded**	**FEs.**	**Succeeded**	**FEs.**	**Succeeded**	**FEs.**	**Succeeded**	**FEs.**
PSO	10	10	55	1	1325	1	24	5	100	1	85
	15	10	53	10	4209	4	77	8	128	4	409
	20	10	48	10	2779	5	96	9	182	5	262
CLPSO	10	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
	15	0	N/A	0	N/A	0	N/A	0	N/A	7	5240
	20	0	N/A	0	N/A	0	N/A	1	238	**8**	3391
CPSO-S_6	10	10	1377	10	555	10	145	10	303	4	319
	15	10	1306	10	450	10	118	10	253	2	138
	20	10	1185	10	398	10	102	10	211	2	116
CPSO-H_6	10	10	1067	10	248	10	61	10	103	2	40
	15	10	1043	10	214	10	56	10	99	2	45
	20	10	992	10	201	10	49	10	101	2	49
$CPSOL_{RP}$	10	10	1149	10	306	10	77	10	143	1	33
	15	10	1145	10	266	10	70	10	144	2	63
	20	**10**	1075	**10**	258	**10**	65	**10**	137	3	116

5.4 Experiment 3: Robustness Analysis

This section compares the various PSOs to determine their relative rankings using both *robustness* and convergence speed as criteria. The term robustness is used here to mean that the algorithm succeeded in reducing the function value below a specified threshold using fewer than the maximum allocated number of function evaluations. A *robust* algorithm is one that can decrease the fitness value below a specified threshold in a fewer number of fitness evaluations during all runs [3].

Table V shows the robustness analysis for all test functions. In function f_0 none of the algorithms, with the exception of CLPSO, have any difficulty reaching the threshold during any of the runs. Table V further shows that all the algorithm solved this problem in fewer than 2000 function evaluations, with the PSO algorithm requiring the fewest function evaluations overall.

The CLPSO algorithm again consistently fails to reach the threshold value of function f_1 on all runs. Furthermore, standard PSO with 10 particles per population is not a robust algorithm, whereas all cooperative algorithms reach the threshold during all runs.

The standard PSO and CLPSO have some difficulty with f_2 function, as can be seen in Table V. The function f_2 represents a very important result regarding the nature of cooperative algorithms: the CPSOLA algorithm may have somewhat slower rates of convergence compared with CPSO-S_6 and CPSO-H_6 algorithms, but it is significantly as robust as them in many cases. The function f_3 shows similar results to function f_2. The cooperative algorithms again perform admirably on the Rastrigin function, but the PSO and CLSPO algorithms are less robust in this test function.

The function f_4 shows interesting results. It proves to be hard to solve for all the algorithms, as can be seen in Table V. None of cooperative algorithms and standard PSO can perfectly optimize it, while CLPSO's novel learning strategy enables it to act more robust when compared to other PSO variants. Also, no algorithm could achieve a perfect score on this test function.

When looking at the number of function evaluations, the CPSO-H_6 algorithm was usually the fastest, followed by the CPSOLA and the CPSO-S_6. These results indicate that there is a tradeoff between the convergence speed and the robustness of the algorithm.

In 4 out of 5 test functions, if the number of fitness evaluations reaches more than 5000 FEs, the algorithm will not meet the robustness condition, while in function f_4 this observation rejects and CLPSO still satisfy the robustness criteria.

5.5 Experiments Analysis

From Table IV, the CPSOLA performs better in 4 out of 5 benchmark functions, where swarm size were 15. This swarm size indicates that the algorithm does not need big population in order to gain better results compared with other PSO variants. Also the CPSOLA performs better while optimizing both unimodal and multimodal optimization benchmarks. Based on Table IV results the algorithm performs better while optimizing complex multimodal optimization benchmarks such as Ackley and Rastrigin. More over from Table V, the proposed algorithm is robust while optimizing complex multimodal functions where there optimization problems have complicated structures.

From the point of complexity analysis, the time complexity standard PSO [2] is $O(g * p * d)$ where n is the number of **g**enerations, m is the number of designated **p**articles and d is the number of **d**imensions. Totally, we assert that the time complexity of PSO is $O(n^3)$.

In this context by considering the Fig. 7, as long as we may have two choices in each generation the time complexity of CPSOLA will be calculated through the following:

$$O(g)*\begin{cases} O(s)*O(s_d)+O(p) \\ O(s)*O(s_d) \end{cases} \quad (7)$$

Where is (7), s is the number of swarms $(s<<p)$ and s_d is the number of member dimensions of s^{th} swarm $(s_d<<d)$. In order to calculate the time complexity of CPSOLA, we consider the maximum of the two terms which is mentioned in (7), thus we have: $O(g * S * Sd + g * p)$. Totally we assert that the time of complexity of CPSOLA is $O(n^{3/S4}+n^2)$. As long as in each generation of CPSOLA algorithm the sub regions of optimization space is optimized through by swarms, the $O(n^3)$ value of PSO is divided by this sub swarms. So, in the worst case that the number of swarms is set to the whole population, the time complexity will be $O(n^3+n^2)$; where this value is worse than the time complexity of standard PSO. But also in the moderate case or best case where the solution space is divided into s_d swarms where $lim\frac{n}{s_d\to\infty}=0$ the algorithm may performs faster than standard PSO with time complexity of $O(n + n^2)$.

6. Conclusions

In this paper we presented a cooperative particle swarm optimizer with adaptive control on the outer layer of cooperation named as CPSOLA. 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 belongs 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 optimized, 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.

To evaluate the performance of CPSOLA, three different kinds of experiments are conducted in this paper. The experiments are carried out on five algorithms on the five chosen test problems belonging to two classes. The CPSOLA utilizes the L_{RP} learning technique as its learning algorithm. The CPSOLA performs the best in unimodal test functions and two out of three multimodal test functions. Totally the CPSOLA is shown the best performance in four out of five test functions. Also, it is significantly a robust algorithm with small standard deviation.

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. MHS '95*, 1995, pp. 39–43.

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

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

[6] 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.

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

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

[9] 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.

[10] 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.

[11] A. Rezvanian and M. R. Meybodi, "LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization," in 3*rd International Conference on Contemporary Computing (IC3 2010), Noida, India. Contemporary Computing*, CCIS, 2010, vol. 94, pp. 64–75.

[12] 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.

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

[14] 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*, Budapest, 2011.

[15] M. Hasanzadeh, M. R. Meybodi, and M. M. Ebadzadeh, "Adaptive Parameter Selection in Comprehensive Learning Particle Swarm Optimizer," presented at the Symposium on Artificial Intelligence and Signal Processing (AISP), Tehran, Iran, 2013, pp. 1–10.
[16] M. Hasanzadeh, M. R. Meybodi, and M. M. Ebadzadeh, "A robust heuristic algorithm for Cooperative Particle Swarm Optimizer: A Learning Automata approach," in *2012 20th Iranian Conference on Electrical Engineering (ICEE)*, 2012, pp. 656 –661.
[17] M. Hasanzadeh, M. R. Meybodi, and M. M. Ebadzadeh, "Adaptive cooperative particle swarm optimizer," *Appl Intell*, vol. 39, no. 2, pp. 397–420, Sep. 2013.
[18] M. Hasanzadeh and M. R. Meybodi, "Deployment of gLite middleware: An E-Science grid infrastructure," in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, 2013, pp. 1–6.
[19] M. Hasanzadeh and M. R. Meybodi, "Grid resource discovery based on distributed learning automata," *Computing*, pp. 1–14.
[20] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature—PPSN III*, pp. 249–257, 1994.
[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in , *IEEE International Conference on Neural Networks, 1995. Proceedings*, 1995, vol. 4, pp. 1942–1948.
[22] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.
[23] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.
[24] M. F. Han, S. H. Liao, J. Y. Chang, and C. T. Lin, "Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems," *Applied Intelligence*, pp. 1–16, 2012.
[25] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.
[26] M. El-Abd, "A cooperative approach to The Artificial Bee Colony algorithm," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–5.
[27] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

Mohammad Hasanzadeh received the B.Sc degree in Software Engineering from South Khorasan Payame Noor University (SKPNU), Birjand, Iran, in 2009 and M.Sc in Artificial Intelligence from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran in 2013. His current research interests include computational intelligence, Machine Learning and Grid Computing.

Mohammad Reza Meybodi received the B.Sc and M.Sc degrees in Economics from Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively. He also received the M.Sc and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.

Mohammad Mehdi Ebadzadeh received the B.Sc in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 1991 and M.Sc in Machine Intelligence and Robotic from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran in 1995 and his Ph.D. in Machine Intelligence and Robotic from Télécom ParisTech, Paris, France in 2004. Currently, he is an Associate Professor in Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. His research interests include evolutionary algorithms, fuzzy systems, neural networks, artificial immune systems and artificial muscles.