Appl Intell
DOI 10.1007/s10489-012-0420-6

Adaptive cooperative particle swarm optimizer

Mohammad Hasanzadeh · Mohammad Reza Meybodi · Mohammad Mehdi Ebadzadeh

Abstract An Adaptive Cooperative Particle Swarm Optimizer (ACPSO) is introduced in this paper, which facilitates cooperation technique through the usage of the Learning Automata (LA) algorithm. The cooperative strategy of ACPSO optimizes the problem collaboratively and evaluates it in different contexts. In the 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. Simulations were conducted on four types of benchmark suites which contain three state-of-the-art numerical optimization benchmark functions in addition to one new set of active coordinate rotated test functions. The results demonstrate the learning ability of ACPSO in finding correlated variables of the search space and also describe how efficiently it can optimize the coordinate rotated multimodal problems, composition functions and high-dimensional multimodal problems.

Keywords Particle Swarm Optimizer (PSO) · Cooperative swarms · Learning automata · Adaptive swarm behavior · Composition benchmark functions · Large-scale optimization · Active coordinate rotated benchmark functions

M. Hasanzadeh (✉) · M.R. Meybodi · M.M. Ebadzadeh
Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
e-mail: mdhassanzd@aut.ac.ir

M.R. Meybodi
e-mail: mmeybodi@aut.ac.ir

M.M. Ebadzadeh
e-mail: ebadzadeh@aut.ac.ir

Published online: 01 March 2013

1 Introduction

Swarm Intelligence (SI) [1] is an abstract system which contains a population of agents interacting locally with each other and with their corresponding environment. SI is inspired from the discipline that lies behind the behavior of flocks of birds, insect colonies and fish schooling. Optimization methods are currently one of the most applicable approaches to solving real-world problems. Particle Swarm Optimizer (PSO) [2, 3] is an optimization algorithm which uses SI as a model of social interactions between multiple agents. PSO [2, 3] optimizes a problem by keeping track of the best values of each individual and the entire population.

Standard PSO [4] is a heuristic-based iterative technique which uses a population of particles. An N-dimensional community is evaluated through an eligibility test in each generation of the algorithm with the best quantities of each individual and the whole population being updated. Another population-based optimization technique is Genetic Algorithm (GA) [5] which uses mating and mutation during evolution as part of the population. The term cooperation was first introduced by Potter in the field of GA [6]. The model consists of three major steps: first; it splits the problem dimensions into smaller parts; second, it solves each of these sub-problems by a single GA; and finally, it combines each subpopulation solution to form an N-dimensional solution vector which is feasible to evaluate through the designated fitness function. The same scenario was brought to PSO technique by Van den Bergh and Engelbrecht [7, 8], including a PSO that splits the solution space into multiple subspaces. The dilemma remains in all cooperative optimization techniques how to select the coherent dimensions in order to solve the problem efficiently.

Learning Automaton (LA) [9, 10] is an autonomous machine which is designed to automatically learn instructions from the environment. The first publication that introduced LA to the science community was a survey conducted by Narendra in 1970s [9]. Since the first introduction of LA, there have been many applications of this learning method such as traffic congestion [11], channel assignment [12] in cellular mobile networks and dynamic point coverage in wireless sensor networks [13]. More recently, LA has been successfully applied to the context of PSO for adaptive parameter selection [14]. Also in [15] a new hybrid model of PSO and cellular automata is developed to address the dynamic optimization.

In order to adaptively select the correlated dimensions of the problem, this paper combines the cooperative PSO [7, 8] with the learning methodology of a group of learning automata [16]. This combination leads to present an optimizer which cooperatively optimizes the problem and intelligently places the correlated dimensions in the same swarm. The performance of ACPSO is evaluated in IEEE Conference on Evolutionary Computation 2005 (CEC 2005) [17] and IEEE Transaction on Evolutionary Computation 2006 (TEC 2006) [18]. Experimental results show that ACPSO algorithm could optimize the rotated multimodal problems better than its counterpart algorithms.

The feature selection [19] is an NP-hard problem and currently has become the focus of much research in areas of application. Considering the fact that we necessarily need specific problems to study the ACPSO correlation detection feature, we have applied it to some active coordinate rotated test functions. The results produced by ACPSO are promising for future design of feature extraction algorithms.

In order to investigate the performance of ACPSO in large-scale optimization problems, we also tested it on six multimodal benchmark functions in 300 dimensions of TEC 2009 [20] and was especially compared with one of the non-cooperative optimization algorithms called Group Search Optimizer [20, 21]. The results indicate that ACPSO could alleviate the curse of dimensionality [22] while optimizing the 300 dimensional problems. The secondary aim of this paper is to provide a brief survey on cooperative optimization techniques, the framework of which are originated from Potter's Cooperative Coevolutionary Genetic Algorithm (CCGA) [6]. Moreover, some applications of learning automata in Evolutionary Algorithm (EA) are reviewed.

The rest of this paper is organized as follows: Standard PSO is introduced in Sect. 2 and then a review of some cooperative algorithms is carried out. In Sect. 3 learning automata, its application in PSO and the technological advances due to its usage are presented. Section 4 describes the adaptive particle swarm optimization. Section 5 is dedicated to empirical study of the proposed method on several test functions and finally in Sect. 6, some conclusions are drawn.

2 Particle swarm optimizer concepts and applications

In PSO [23] a number of particles fly in the problem space and each of them evaluates its current position through the designated objective function. Then, each particle calculates its move by combining its own best information and best information of all the swarm members. PSO has been used in a wide range of applications including bandwidth minimization problem [24], distributed local area networks [25], e-learning systems [26] and armored vehicle design [27].

2.1 Conventional particle swarm optimizer

Assume that an N-dimensional problem exists, and the corresponding PSO contains a population of N-dimensional particles. Each particle indicates a feasible solution in the search space. Three features are assigned to the ith particle of the population: the position vector $X_i = (x_i^1, x_i^2, \ldots, x_i^N)$, the velocity vector $V_i = (v_i^1, v_i^2, \ldots, v_i^N)$ and the best position which is met $X_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^N)$. The velocity V_i^D and position X_i^D of the ith particle are updated through the following equations [4, 18]:

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

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

where $gbest = (gbest^1, gbest^2, \ldots, gbest^D)$ is the best position which is met by the whole population, c_1 and c_2 are acceleration constants in which they control the absorption degree of *pbest* and *gbest* positions, also *rand1* and *rand2* $\in [0, 1]$ are two random numbers and finally w [28] is called inertia weight which is designed to balance the exploration and exploitation characteristics of PSO. The algorithm of the original PSO is given in Algorithm 1.

2.2 Some population-based optimization approaches

Since introducing PSO for the first time [3], there has been a great deal of research on improving the performance of the original PSO. Comprehensive Learning Particle Swarm Optimizer (CLPSO) [18] is a PSO with a new strategy. The main goal of CLPSO is to avoid premature convergence when solving multimodal problems. In the new strategy, each particle learns information about all particles from *pbest*. The velocity of particles in each dimension is updated by an exemplar function. The exemplar function collects *pbest* of all particles and assigns an exemplar to each dimension of them using a tournament selection. Since this novel learning schema maintains the diversity of the population, CLPSO could avoid premature convergence.

The Group Search Optimizer (GSO) [20, 21] is an optimization heuristic which adopts the Producer-Scrounger

Algorithm 1 Standard PSO

```
for each generation do
    for each individual $i$ in the population do
        update position of $i$th individual:
        $V_i^D = w \times V_i^D + c_1 \times rand1_i^D \times (pbest_i^D - X_i^D) + c_2 \times rand2_i^D \times (gbest^D - X_i^D)$
        $X_i^D = X_i^D + V_i^D$
        calculate individual fitness $f(x_i)$
        update $pbest_i$ and $gbest$
    end for
end for
```

(PS) framework. The PS model is a group living methodology with two strategies: (1) producing, e.g., searching for food, and (2) joining (scrounging), e.g., joining resources uncovered by others. Besides producer and scrounger members, the population of GSO algorithm also contains some additional dispersed members who perform random walks to avoid getting trapped in pseudominima.

The original PSO doesn't offer a fast convergence speed and is thus easily trapped in local optima. To obtain the abovementioned goals, Adaptive Particle Swarm Optimization (APSO) is proposed in [29]. APSO includes two main phases. First, observing the population distribution and computing the evolutionary factor f by the distance of particles. Based on the information given by f, one of the four evolutionary states, namely *exploration*, *exploitation*, *convergence* and *jumping out* will be chosen for the next generation. These evolutionary strategies are developed to adaptively control the acceleration coefficients (c_1 and c_2). Moreover, APSO dynamically adjusts the inertia weight w. Second, if the selected state was convergence, an elitist learning would be performed. Similar to Simulated Annealing (SA), elitist learning chooses one dimension of *gbest* and changes it by a Gaussian perturbation.

DNA sequence compression algorithm [30] is an approach that tries to find the most effective way of data encoding to reduce the space needed to store the data. In [30], a novel Adaptive Particle Swarm Optimization-based Memetic Algorithm (POMA) is proposed for DNA sequence compression. After computing the codebooks, each codebook is encoded into a particle. Then, in order to find the optimal codebook, the particles are given to the CLPSO algorithm [18] which performs a global search over the population. Finally, Adaptive Intelligence Single Particle Optimizer (AdpISPO) is run, which implements the search with only one particle and performs a local search. AdpISPO is an adaptive version of ISPO [31] which splits the solution vector into a certain number of subvectors and updates them sequentially. The key characteristic of ISPO is its learning factor that intelligently tunes the particle velocity during the search process. By combining these two PSO algorithms, POMA keeps track of global and local search strategies promisingly.

Due to the shortcomings of conventional PSO, an Orthogonal Learning Particle Swarm Optimization (OLPSO) is recommended in [32]. Similar to CLPSO [18], the Orthogonal Learning (OL) strategy tries to construct a direction-efficient exemplar. There are three motivations for a typical OL strategy: First, building an accurate guidance vector based on search information from the best personal and neighborhood positions of the particles. Second, applying Orthogonal Experimental Design (OED) [33] to construct a beneficial learning exemplar. Third, using the notion of CLPSO learning strategy [18] to develop OL strategy which can determine the search direction accurately.

Light Adaptive PSO (LADPSO) [34] is a PSO in which fuzzy logic is utilized to improve the standard PSO. LADPSO adds two operators to PSO, a plow operator for efficient initialization and a mutation operator to avoid getting trapped in local minima. The combination of these two operators facilitates both global and local searches of LADPSO.

A PSO is designed in [35] for dynamic environments. Furthermore, master-slave architecture is applied to PSO algorithm in [36]. Each of the slave swarms evolve independently while the master swarm combines its own information with the information received from the slave swarms in order to evolve more efficiently. A multi swarm cooperative PSO with four subswarms is introduced in [37]. In this PSO algorithm, various techniques are employed to maintain the diversity of the population, escape from pseudominima and acquire a better solution. In [38] a novel PSO based on emotional behavior is proposed to solve real optimization problems. In this approach, an emotional factor splits up the search space into potential regions that are finely explored by subswarms.

2.3 Review of cooperative optimization heuristics

The first attempt to develop cooperative optimization was based on Genetic Algorithms (GA) [6]. The Genetic Algorithm (GA) [5] is one of the basic branches of Evolutionary Computing (EC) which was founded by Holland. Particle

Swarm Optimization (PSO) [4] is a newfound optimization method which has a lot of applications in spite of its simple structure. For solving multimodal problems or functions which cannot be solved by other heuristics, PSO offers more promising results than other methods. The current publication trend marks the usage of this method in rather new and different applications. The main categories of PSO applications include video analyzing, design and reconstruction of electronic networks, control applications, scheduling applications, biological, medical and pharmacy applications [23].

Having been inspired from natural evolution flow, GA employs recombination, selection and mutation operators to produce an optimal set or entity from its early population. In this context, Potter considered a typical solution of the optimization problem as an N-dimensional vector where each dimension represents a subpopulation of the primary population. In order to evaluate one member of a subpopulation, an N-dimensional vector should be constructed by combining the other selected members of each subpopulation. Similar to the standard GA, the constructed vector could easily be evaluated through the designated objective function. The Cooperative Coevolutionary Genetic Algorithm (CCGA) of Potter is sensitive to the correlation of dimensions and its performance deteriorates upon optimizing correlated variables of the optimization problems. Ong, Keane and Nair [39] raised the idea of Potter's cooperative GA by Radial Basis Function (RBF) and used it to solve some problems of the correlated parameters.

A Cooperative PSO (CPSO) was first introduced by Van den Bergh and Engelbrecht in [7]. The CPSO algorithm splits the input vector into several subvectors and optimizes them using the assigned subpopulation. In CPSO, each swarm acts as the representative of one to several dimensions of the problem while the cooperative swarms collaboratively optimize the problem solution. Van den Bergh utilized these subpopulations for training the neural network (CPSO-S). He also used this cooperative approach in function optimization and eventually introduced Hybrid Cooperative PSO (CPSO-H) [8]. The CPSO-H algorithm is a cooperative PSO which combines standard features of PSO with the cooperative behavior of CPSO by the aim of utilizing both beneficial characteristics of the aforementioned PSOs.

Evolutionary Strategy (ES) [40] is introduced by Rechenberg and Schwefel in the 1970s in Germany. At that time, the applications of ES were limited to optimizing the nasal shape. Sofge, De Jong, and Schultz [41] brought the cooperation concept into ES. The Cooperative Coevolution Evolutionary Strategy (CCES) divides the population of ES into some subspecies and lets them evolve. By means of a migration operator, Sofge could hybridize the cooperative evolutionary behavior of Potter with ES. The proposed model controls the interaction of subspecies properly and exhibits good performance results.

Differential Evolution (DE) [42] is a population-based parallel search method which is used for global optimization problems. In each generation of this algorithm, the population moves toward the global optimum by mutation, cross over and selection operators. Shi, Teng and Li [43] applied the cooperative behavior of Potter to DE and invented Cooperative Co-evolutionary Differential Evolution (CCDE). This CCDE fragmented the standard problem into several subproblems and allocated a subpopulation to each of them. Yang, Tang and Yao [44] introduced a randomized grouping mechanism and used an adaptive weighting strategy in order to adapt the separated components. The idea was to bring the interacted variables into a similar subcomponent. Later, Yang introduced a self-adaptive neighborhood search into DE (SaNSDE), which could tackle the non-separable problems with more than 1000 dimensions inside.

Artificial Bee Colony (ABC) [45], which is inspired from natural bee colonies, is another swarm intelligence method. El-Abd [46] exerted the cooperative approach of Potter into ABC and produced Cooperative Artificial Bee Colony (CABC). 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.

The discussed cooperative issues assert that the cooperative coevolutionary approach is implemented in several EAs, among which the following cooperative techniques are reviewed: Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [2], Evolutionary Strategy (ES) [40], Differential Evolution (DE) [42] and Artificial Bee Colony (ABC) [45].

2.4 Comprehensive study of particle swarm optimizer

Standard PSO [4] and cooperative PSO [8] were introduced in Sects. 2.1 and 2.3, respectively. The key entity of these optimization approaches 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 flies through an N-dimensional search space by N values corresponded to each dimension of the space*". To understand this phrase, consider the population as a matrix $[M \times N]$ where M and N respectively represent the number of particles and dimensions as [# *of Particles* $\times$ # *of Dimensions*] (see Fig. 1). In this framework, velocity and position of the standard PSO [4] population were updated row-wise. The

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

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

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

Fig. 1 Comprehensive view of the PSO population. $f(P_i)$ represents the ith particle of population which evaluates through the traditional PSO mechanism and $f(P_i, S_j)$ indicates the evaluation process of the ith particle (P_i) of jth swarm (S_j) of CPSO population

interpretation of this framework in CPSO [8] is quite different from that of the standard PSO algorithm. In CPSO algorithm, 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 the corresponding swarm and the best particles of other swarms.

3 Learning automata

Learning Automaton (LA) [16] is a typical model of observation which adapts itself to the dynamic environment. The learning process of an automaton is acquired and applied simultaneously. The concept of learning systems is an inevitable phenomenon which has rapidly grown in the last decades. This emergence of intelligent systems is due to the ease of learning tasks, which tends to replace the manual adjustment of chores and applications simply and automatically. The following will review some interesting and new applications of LA.

To attain high coverage in wireless sensor networks, an automatic node deployment is proposed in [47]. For solving vertex coloring problem, Torkestani and Meybodi proposed an algorithm in [48]. Moreover, learning automaton was successfully applied to the Traveling Tournament Problem in [49].

Several methods have been proposed for using learning automata in Evolutionary Computing (EC). For the Differential Evolutionary (DE) algorithm, Noroozi, Hashemi and Meybodi invented an optimization model which combines the DE algorithm with Cellular Automata (CA) for dynamic environments [50]. Meanwhile, a CA splits the problem space into cells with the subpopulation placed at each cell in CellularDE [50]. Vafashoar, Meybodi and Momeni proposed another DE variant based on learning automata. Their proposed method called CLA-DE algorithm [51] combines Cellular Learning Automata (CLA) with differential evolution. CLA-DE iteratively partitions the dimensions of the

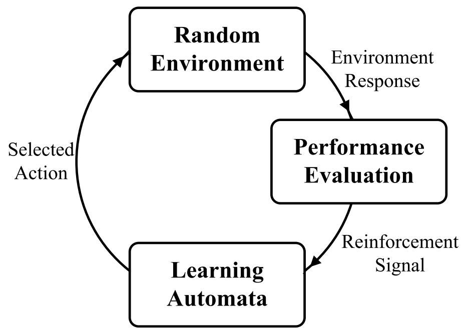

Fig. 2 Visualization of a learning automaton in a stochastic circumstance

search problem and learns the most promising regions of the corresponding dimensions. In order to avoid the population diversity loss, a CA is used in [52] to evaluate the structure of the genetic algorithm.

Artificial Immune System (AIS) [53] is one of the computational intelligence branches which inspired from natural immune system. In [56] a new method of AIS utilizing the cooperative concept of learning automata is proposed. The firefly algorithm [54] is a new evolutionary optimization algorithm which works based on the flashing characteristics of fireflies. In [55], parameter adaption of firefly algorithm is handled by learning automata.

Below explains the operation of LA in more details. Learning automata [9–11] are stochastic learning machines which gradually adapt with the corresponding environment. An automaton contains a set of actions every time the automaton interacts with the environment. Then, one of the actions is selected and the environment sends a reinforcement signal as feedback. Eventually the automaton could recognize whether its selected action is wrong and update its action probability vector. A schematic representation of leaning automaton is depicted in Fig. 2.

The objective of a learning automaton is to grasp an optimal choice from a random environment. This perception is not acquired, unless the automaton randomly selects an action and little by little learns the optimal action. *Variable-structure automatons* are a typical example of stochastic systems. A Variable-Structure Learning Automaton (VLSA) is identified by a quadruple $[p, \alpha, \beta, T]$, where $p(n+1) = T[p(n), \alpha(n), \beta(n)]$ is the reinforcement scheme of the automaton. If $p(n+1)$ is a linear function of $p(n)$ the schema is called linear, otherwise it is non-linear. In the noted learning scheme, T is the learning algorithm, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ is the set of inputs and $p = \{p_1, p_2, \ldots, p_r\}$ is the probability of each action. The pseudo code of a variable structure learning automaton in a stationary environment with $\beta \in \{0, 1\}$ and r actions is shown in Algorithm 2.

In (3) and (4), a and b are called *learning parameters* and they are associated with the reward and penalty responses.

Algorithm 2 Learning automata probability vector update framework

```
define
Initialize $r$-dimensional action set: $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ where $r$ is the number of actions.
Initialize $r$-dimensional action probability vector: $p = \{(\frac{1}{r})_1, (\frac{1}{r})_2, \dots, (\frac{1}{r})_r\}$
while (the automaton converge to one of its action)
  The learning automaton selects an action based on the probability distribution of $p$.
  The environment evaluates the action and calculates the reinforcement signal $\beta \in \{0, 1\}$.
  The environment feedbacks $\beta$ to the learning automaton.
  Consider $i$ as the selected action of the automaton, $j$ as the current checking action and $n$ as the $n$th
  step of evolution.
  Update the probability vector:
  for each action $j \in [1, \dots, r]$ do
    if $\beta = 0$        // positive response

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

    else if $\beta = 1$        // negative response

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

    end if
  end for
end while
```

Considering the values of a and b, there are three types of learning algorithms. In Linear Reward-Penalty algorithm ($L_{R\text{-}P}$), it is considered that a and b are equal. In Linear Reward-Inaction ($L_{R\text{-}I}$) the learning parameter b is set to 0. And finally in Linear Reward-epsilon-Penalty ($L_{R\varepsilon P}$) the learning parameter b is much smaller than a.

3.1 Improved PSOs using learning automata

Parameter adaption [14, 55, 57] is a successful application of learning automata. Due to sensitive parameters of PSO, Hashemi and Meybodi adaptively adjusted the values of inertia weight and acceleration coefficients of PSO at two levels of the population, swarm degree and particle degree [14]. In order to address the diversity loss of the PSO population in dynamic environment, a multi swarm PSO has been put forward in [15]. This Cellular PSO [15] used an N-dimensional CA with C^D cells in an N-dimensional problem space. The developed method contributed to save the population diversity.

Rastegar, Meybodi and Badie designed a LA-based Discrete PSO (DPSO) algorithm [58], which benefits from a collection of LA to comprehend the topological space of the optimization problem. Each LA is assigned to one dimension of the particle and can be viewed as the controller of it. The Bitwise LA which was used in the DPSO algorithm estimates the particles position and moves them through the search space. Jafarpour, Meybodi and Shiry [59] enhanced the LA-based DPSO using a CLA neighborhood topology, in which each particle was placed in a cell being affected by its best personal information and best information of neighboring particles. Learning automaton has greatly influenced the way in which particles fly in the search space.

There have been lots of publications on trend and usage of learning in particle swarm optimization. The new model PSO-LA by Sheybani and Meybodi [60] introduced the usage of LA for path and velocity control of PSO in real parameter optimization. An application of PSO-LA in sensor network is addressed in [61]. PSO-LA used one LA to determine the trajectories of particles in PSO. Other efforts toward using LA to control the behavior of the PSO population have been described in [62]. In order to balance the process of global and local searches in this new PSO-LA algorithm, one learning automaton is assigned to each particle of the swarm. A recent advancement in combining learning automata with PSO was Dynamic PSO-LA (DPSOLA) [63]. The proposed model used three types of existing information, namely individual, neighboring and swarm information. In order to confirm the velocity update equation of

Algorithm 3 Cooperative evolutionary algorithm framework

```
define
Split $N$-dimensional search space into $j$ subpopulations of entities.
Calculate the best individual of each subpopulation ($sbest$).
Construct a Context Vector (CV) through the best individuals of each subpopulation:
$\mathbf{CV} = [sbest_1, sbest_2, \ldots, sbest_j]$
for each generation $i$ do
  for each subpopulation $j$ do
    for each entity $k$ do
      Replace current entity of the $j$th subpopulation by its corresponding positions in the CV
      Evaluate the $N$-dimensional output vector through the fitness function.
      $k = k + 1$ // next entity
    end for
    Apply cooperative behavior of EA to $j$th subpopulation.
    Update $sbest_j$.
    $j = j + 1$ // next swarm
  end for
  $i = i + 1$ // next generation
end for
```

PSO, LA selected a combination of the aforementioned information. The new algorithm had the ability to escape from the pseudominima and converge quickly. Another hybrid algorithm based on PSO and learning automata was Cooperative PSO-LA (CPSOLA) [64] which improved the performance of CPSO-H [8] using a learning automaton as an adaptive switching mechanism.

4 Adaptive cooperative particle swarm optimizer

Cooperative PSO [7, 8] 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 prime solution of each subpopulation within an N-dimensional vector. Typically to evaluate the current subpopulation, the corresponding dimensions filled with the position of the particle and the other dimensions are considered constant. The following is a typical cooperative coevolutionary pseudocode (Algorithm 3).

Finding correlated variables is an important application of cooperative approaches. Two variables are correlated since changing one of them shows an impact on the other one. A simple example of two correlated variables is the relationship between weight and height of people. If the height of someone is more than the average value, it is usually expected that his/her weight is also above the average. *Correlation* is a measure for calculating the power of association between two variables. However, *covariance* is a measure of calculating the correlation between each pair of correlated variables. In standard benchmark functions which are used in [6, 7, 41, 43, 46], all dimensions are independent from each other, but in coordinate rotated test functions which are introduced in [8, 18], dimensions of the problem are correlated. The rotation matrix, which is used to rotate the variables of the problem, correlates all dimensions of the search space. The correlation behind sub dimensions is a reason to divide the search space and optimize each division by a subpopulation. However, all dimensions of the standard coordinate rotated benchmark functions are correlated together. What if one changes the benchmark consciously? If one rotates the objective function so that the correlation lies exactly behind some specific dimensions of the problem, he/she will have the chance to allocate isolated subpopulations into the correlated sub dimensions. This kind of credit assignment to swarms will eventually lead to improved performance of the algorithm and make the test function more realistic. In order to show the impact of correlated variables on the performance of PSO, several active coordinate rotated test functions are designed; the sub dimensions of which are affected by the rotation.

The benchmark functions which are used to evaluate the cooperative PSO [7, 8] and other cooperative coevolutionary algorithms [6, 41, 43, 46] evaluated the dimensions of the problem independently. There are three drawbacks in these cooperative approaches. First of all, neglecting the correlation between variables by a fully coordinate rotation

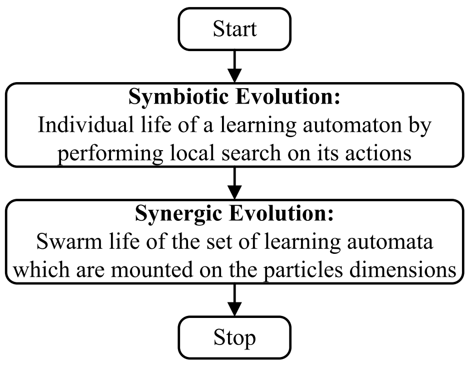
Start
Symbiotic Evolution:
Individual life of a learning automaton by performing local search on its actions
Synergic Evolution:
Swarm life of the set of learning automata which are mounted on the particles dimensions
Stop

Fig. 3 Two key steps of the ACPSO algorithm

matrix which is used to rotate the problem dimensions is one major problem. Second, there is no term of correlation or covariance considered in their updating schema. And finally, the lack of a proper selection mechanism for the correlated sub dimensions can consequently lead to the selection of correlated dimensions unadvisedly in a random or fixed fashion. Originally, these algorithms do not see the correlation of variables and they are not suitable for real life optimization problems such as wireless communications, nuclear science, signal processing, etc. For these kinds of applications an alternative approach is needed which intelligently selects the correlated variables and optimizes them in mutual cooperation manner. In addition to accelerate the optimization process, optimizing correlated variables together could also make reaching the problem optimum more accurate.

Based on the previous discussions, Adaptive Cooperative Particle Swarm Optimizer (ACPSO) is proposed here. The key idea of ACPSO algorithm is to find the correlated variables of the search space and integrate them into a joint swarm. A set of learning automata is used to extract the correlated dimensions. A learning automaton is an autonomous machine with finite number of actions. The automaton is originally like an inhabitant of stochastic ecosystem which interacts with the environment during the evolution part of its life. The responses of environment to the automaton gradually leads to the emergence of a survival behavior for the automaton.

A typical ACPSO algorithm consists of two phases: *symbiosis* and *synergy*. In the symbiosis stage, the learning automata are allowed to learn the correlated parameters, while in synergy stage, these correlated dimensions are put in the same swarm and are optimized directly by CPSO algorithm. Also in the synergy phase, the selected dimensions of each group may vary such that the learning automata still have the chance to learn the correlation. The schematic view of ACPSO algorithm is illustrated in Fig. 3.

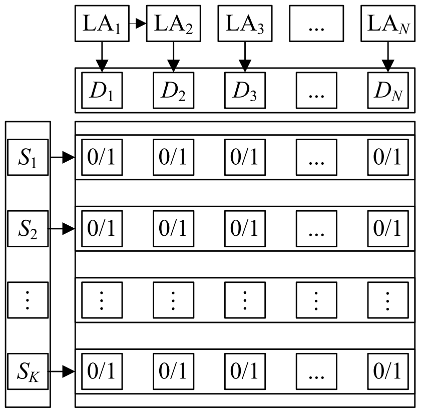

Fig. 4 Binary swarm table and the embedded learning automaton for each dimension. $S = \{S_1, \ldots, S_K\}$ indicates the swarm set, $D = \{D_1, \ldots, D_N\}$ indicates the problem dimensions and $\mathrm{LA} = \{\mathrm{LA}_1, \ldots, \mathrm{LA}_N\}$ represents the learning automaton set which are allocated to each dimension

4.1 Introducing the binary swarm table

The fundamental data structure of ACPSO algorithm is a swarm table. The swarm table characterizes which dimensions belong to which swarm. A schematic view of this table is shown in Fig. 4. Each row of the swarm table represents a swarm and each column represents a dimension of the search problem. In Fig. 4, K denotes the number of predefined swarms and N is the dimensions of the problem. Having assigned the dimensions to the swarms, the corresponding dimensions of each swarm will be set to 1 and others to 0. Note that each dimension can be only placed in one swarm.

4.2 Embedding learning automata in the swarm table

In order to assign problem dimensions to swarms, a learning automaton was placed upon each dimension ($|\mathrm{LA}| = |D| = N$ where, '$||$' indicates the cardinality of a set, $D = \{D_1, \ldots, D_N\}$ is the set of problem dimensions with $|D| = N$ and $|\mathrm{LA}|$ gives the cardinality of LA set). The placed learning automata have total number of actions equal to the initial number of swarms in PSO population ($r = |S| = K$ where, $S = \{S_1, \ldots, S_K\}$ and K denotes the number of swarms ($|S| = K$)). The initial probability of each action is $1/r$ (where r is the number of actions). Figure 5 presents the assignment of learning automata to the swarm table and the mapping procedure of swarm table to the population. Each learning automaton is mounted on a particular dimension of the search space. There are two basic tasks of the learning automata: action selection and swarm dimension assignment; each automaton should perform them in each trial. At first, a context vector is required to evaluate the particles of the population. For simplicity in Fig. 5 the context vector is formed by concatenating the global best position (*gbest*) of each dimension.

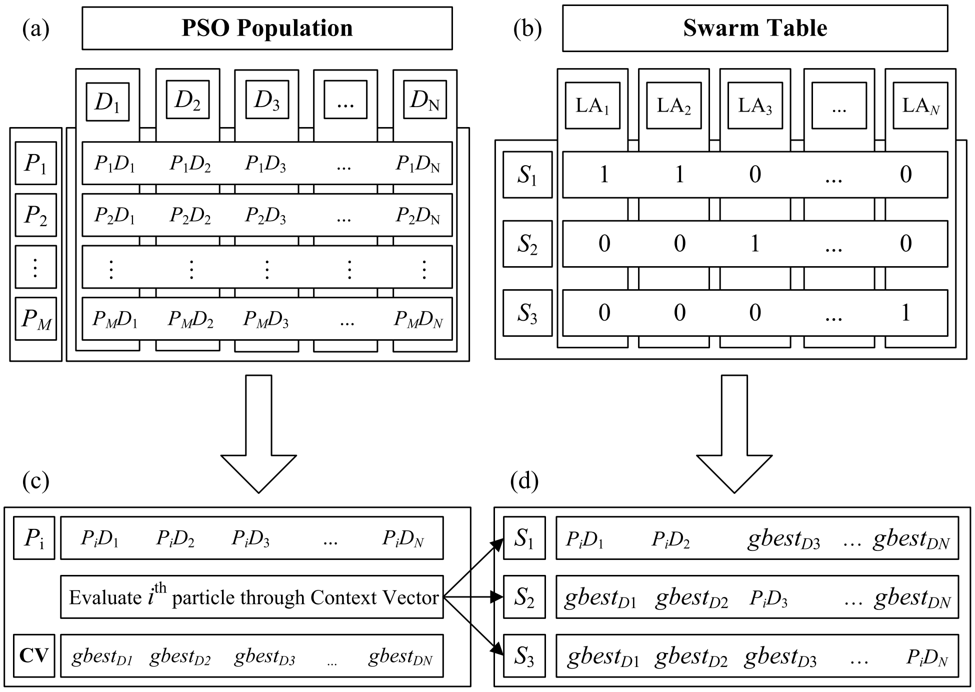

Fig. 5 Mapping process between PSO population and swarm table. The example consists of M particles with N dimensions: $P = \{P_1, \dots, P_M\} | P_i = \{P_i D_1, \dots, P_i D_N\}$ (**a**). Typically the number of swarms is set to 3. In first step the learning automata select their actions and scatter the dimensions into the swarms. After selecting the action, corresponding dimensions of the swarm will be marked as 1 in the swarm table (**b**). Moreover, if we want to evaluate the ith member of the jth swarm, we will look up into the jth row of swarm table (**c**). To form an N-dimensional solution vector and calculate the fitness for the ith particle of jth swarm the components with 0 in the jth row are remained constant as their values in the context vector (**CV**), while the components with 1 are replaced by corresponding values of the ith particle (**d**)

Note that during further development of the ACPSO algorithm, the context vector will be constructed from the best position of the swarm (*sbest*) where each *sbest* is not a sequential subset of discussed context vector from this section. Finally the particles of a swarm will evaluate through the context vector based on the 0/1 values of the swarm table. This mapping procedure is a part of the ACPSO framework which will be discussed in detail in the following section.

4.3 ACPSO framework

Adaptive cooperative PSO is a learning, cooperative and optimization algorithm. To simultaneously gather all these aspects together, one should establish a collective behavior between the PSO population and a set of learning automata mounted on each dimension. A *fuser* module is first needed to match the learning ability of a set of the learning automata to the PSO population. A fuser gathers all information from the PSO population and computes a reinforcement signal for the learning automata. The schematic of such a module is shown in Fig. 6. The PSO population is the environment from which the learning automata should learn from. The action set for each learning automaton is $\alpha = \{\alpha_1, \dots, \alpha_r\}$; $r = K$. Also the action probability vectors are associated with each learning automaton of the fuser module. Each LA of the fuser module selects an action based on its action probability vector. The learning automata correlate with the swarm table and the selected action maps into it. Each particle of the current swarm evaluates through the context vector and the fitness of the related swarm spreads to the fuser module as reinforcement signals. The fuser combines these reinforcement signals and obtains the reinforcement signal for all learning automata of the swarm. For each automaton of the swarm, the action probability vector is updated according to its previous action and its received reinforcement signal.

The proposed algorithm consists of two steps nominated as symbiosis and synergy. The following will explain them more clearly.

4.3.1 ACPSO symbiosis step

The aim of the symbiosis step is to give the set of LA the required time to learn the correlated variables of the search space before the algorithm starts to optimize the problem intentionally. At first, the learning automata select their actions

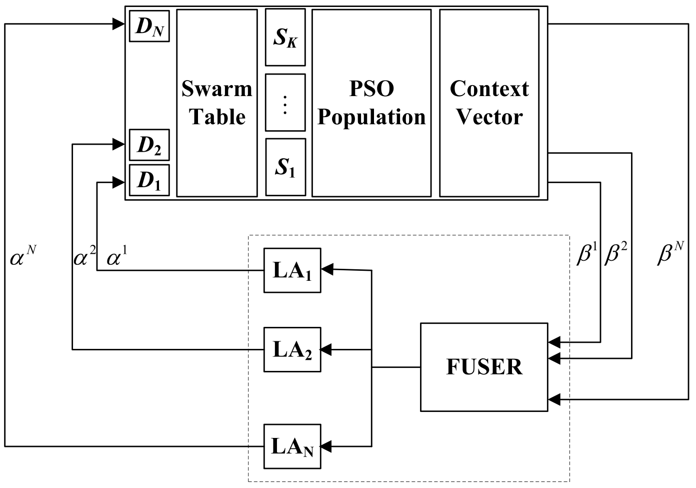

Fig. 6 Module of learning automata

Procedure 1 Initialize context vector

```
define
Number of swarms $K$ and number of dimensions $N$.
Let $S = \{S_1, \ldots, S_K\}$ denotes the set of swarms.
Let $D = \{D_1, \ldots, D_N\}$ denotes the set of problem dimensions.
Let $A(d) \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in corresponding $d$ positions of vector $A$. Also, let
$A \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in vector $A$.
Let $x$ denotes PSO population.
Let $gbest = [gbest_1, gbest_2, \ldots, gbest_N]$ denotes the global best position of initial population.
Initialize CV by global best position of population: $\mathbf{CV} = gbest$
begin
  for each swarm $j \in [1, \ldots, K]$
    Find $S_j$ swarm members:
      Let $d$ be the set of corresponding dimensions of $S_j$:
        $d \subseteq D \mid$ SWARM_TABLE$(j, d) = 1$
    for each particle $i \in [1, \ldots, PS]$
      Replace $d$ positions of CV with corresponding values in $x_i$: $\mathbf{CV}(d) \leftarrow x_i(d)$
      if fitness $(\mathbf{CV}) <$ fitness $(gbest)$
        $sbest_j \leftarrow x_i(d)$
      end if
    end for
  end for
  Let $\mathbf{CV} = [sbest_1, sbest_2, \ldots, sbest_K]$ be the context vector.
end
```

based on their initial probability vectors which are equal to 1 divided by the number of actions (as mentioned before the action set size of each automaton is equal to the number of swarms). After filling the swarm table, the particles of each group are evaluated using a context vector. Note that a context vector is constructed from the swarm best positions of each swarm (*sbest*). As mentioned, the basic goal of the symbiosis step is to let the learning automata grasp a comprehensive view about the correlation of variables. The following procedures describe the details of this step. Note that there are commonalities with the symbiosis step.

Initialize context vector In order to calculate each swarm best position (*sbest*) and concatenate them to form the context vector, this procedure will be used at the beginning of the ACPSO algorithm. At first, corresponding dimensions of each swarm is extracted from the swarm table. Then the best particle among each subpopulation is chosen and considered as *sbest*. Concatenation of these particles leads to form the

Procedure 2 Context vector (d, p)

```
define
Current particle $p$, number of swarms $K$ and number of dimensions $N$.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms.
Let $D = \{D_1, \dots, D_N\}$ denotes the set of problem dimensions.
Let $d \subseteq D$ be a subset of $D$.
Let $A(d) \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in the corresponding $d$ positions of vector $A$. Also, let
$A \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in vector $A$.
Build CV from all swarms best position: $\mathbf{CV} = [sbest_1, sbest_2, \dots, sbest_K]$
begin
  Replace $d$ positions of CV with the corresponding values in $p$: $\mathbf{CV}(d) \leftarrow p(d)$
  Return CV.
end
```

Procedure 3 Action refinement (i)

```
define
Current generation $i$, number of dimensions $N$ and swarm number $K$.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms.
Let $D = \{D_1, \dots, D_N\}$ denotes the set of problem dimensions.
Let $\mathrm{LA} = \{\mathrm{LA}_1, \dots, \mathrm{LA}_N\}$ denotes the set of learning automaton designated to each dimension.
begin
  Calculate LA index for refinement:
    $d = (i \bmod N) + 1 \mid d \in D$
  Select random swarm $j$:
    $j \in [1, \dots, K] \mid \mathrm{SWARM\_TABLE}(j, d) \neq 1$
  Change the selected action of $\mathrm{LA}_d$ to $j$.
  Let $V = \{V_1, \dots, V_q\}$ denotes a subset of swarms set $S$.
  Let $V = \{S_1, \dots, S_K\} - S_j \mid V \subset S$ be the subset of $S$ except for $S_j$, while $q = K - 1$.
  Update swarm table:
    Fill $j$th row of swarm table with 1:
      $\mathrm{SWARM\_TABLE}(j, d) = 1$
    Fill $V$ row members of swarm table with 0:
      $\mathrm{SWARM\_TABLE}(V_l, d) = 0 \mid l \in [1, \dots, q]$
end
```

context vector. The discussed procedure is illustrated in Procedure 1.

Context vector In cooperative approaches, the global best particle of the population includes the *sbest* information of each swarm, while the information composition of these *sbests* would construct an N-dimensional context vector. While evaluating the particles of jth swarm, the context vector function returns an N-dimensional vector which consists of all the global best particles in all swarms, except for jth swarm, which is replaced with the position of any particle of jth swarm. Procedure 2 presents the context vector subroutine.

Action refinement In this subroutine and after completing each evolution round of the PSO population, one of the swarm table dimensions will be chosen sequentially and its corresponding learning automaton will change its action randomly. In the next generation, the impact of this change is evaluated in the swarm table with the isolate update procedure being applied to the corresponding learning automaton. This procedure performs a local search on the learning automaton action set and lets the reward and penalty signals be more effective. Besides giving vague reinforcement signals to the learning automata from the beginning, the LA is allowed to identify the correlated variables more preciously and gradually. Procedure 3 depicts pseudocode of the action refinement procedure.

Reinforcement signal The learning automata should detect a signal from the environment regarding the impact of their actions. Having a suitable feedback from the population of

Procedure 4 Rei nforcement signal (t, j)

```
define
Improvement tag $t$ and current swarm $j$.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms.
Let $sbest_j \mid j \in [1, \dots, K]$ denotes the swarm best position of $S_j$.
begin
  if $t == 1$
    Reinforcement Signal $= 0$ // $gbest$ is improved by $S_j$ or $sbest_j$ is improved.
  else if $t == 0$
    Reinforcement Signal $= 1$ // $gbest$ is not improved by $S_j$ or $sbest_j$ is not improved.
  end if
  $signal$ = Reinforcement Signal
  Return $signal$.
end
```

swarms and fusing it to convey an operational token will play an important role in building an efficient reinforcement signal. In order to construct a suitable reinforcement signal, several ways are examined such as: mean improvement on *pbest*'s information of the particles, mean *gbest* improvement of the population and *gbest* improvement of the population. This observation implies that the signal made from *gbest* information of the PSO population is the most effective one and also takes into consideration fairness about the population. The reinforcement signal shows the learning automata whether their actions are right or wrong. The signal is defined by (5) below:

$$\textit{Reinforcement Signal} = \begin{cases} 0 & \textit{fitness}(gbest^i) < \textit{fitness}(gbest^{i-1}) \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where i is the current iteration number. As all test problems are a kind of minimization function, the smaller their fitness, the better they are. If the fitness of the global best particle of the population (*gbest*) is improved (decreased) by a specific swarm, then the corresponding swarm will get the reward signal. Since the *gbest* improvement is equal to or greater than that of the swarm best position (*sbest*), one can additionally define the reinforcement signal via the *sbest* information of each swarm. This schema is executed after evaluating each swarm of the PSO population. The signal is only applied to a learning automaton (in the isolate update procedure) or a set of LA that are associated with the current swarm (in the ensemble update procedure). Then, the corresponding probability vectors of the LA are modified based on the learning algorithm of automata using (3) and (4). Procedure of the reinforcement signal is illustrated in Procedure 4.

Parameter t is obtained from the body of the ACPSO algorithm. Also the result of the reinforcement signal procedure will be coupled with the isolate update and ensemble update procedures in symbiosis and synergy steps, respectively.

Isolate update If someone wants to apply the reward and punishment signal in just one dimension, he/she will use the isolate update procedure (see Procedure 5). The function uses the reinforcement signal which is calculated for the current dimension and lets the learning automaton independently perform its update task on the corresponding dimension (the reinforcement signal will be defined later). The isolate update procedure will be further developed in the synergy step, which has additional applications in the swarm table.

4.3.2 ACPSO synergy step

After selecting dimensions in the symbiosis step, which would place the correlated subsets of variables in the same swarms, the synergy step is initiated. Till now, the swarm table may assign wrong or incomplete dimensions to the swarms and these noisy swarms may deteriorate the fitness value during evaluation. The aim of this phase is to exploit the correlation detection power of the set of learning automata and also to optimize the test function explicitly.

Similar to the symbiosis interval, the swarms contribute to the context vector and each particle optimizes the problem through different contexts. As discussed in the isolate update procedure, the reinforcement signal leads to find the best policy of the automaton. In other words, the reinforcement signal helps the automaton to identify a subset of correlated variables in the search space. The swarms contain a collection of learning automata and the reinforcement signal should be provided for all of them. Thus, a procedure is required to apply the reinforcement signal for a bunch of LA.

Finally, the ACPSO algorithm needs a function to initialize the swarm table at the end of each generation. This mechanism is performed based on the probability vector of each automaton. The following procedures sketch work flow of the synergic step:

Procedure 5 Isolate update (*signal*, *j*)

```
define
Reinforcement signal signal, current swarm $j$, current dimension $i$, number of swarms $K$ and number of dimensions $N$.
Let $\mathrm{LA} = \{\mathrm{LA}_1, \dots, \mathrm{LA}_N\}$ denotes the set of learning automata designated to each dimension.
Let $\mathrm{LA}_i \mid i \in [1, \dots, N]$ denotes the selected automaton for updating its probability vector by input signal.
Let $\alpha = \{\alpha_1, \dots, \alpha_r\}$ denotes the action set of $\mathrm{LA}_i$, where $r = K$.
Let $\alpha_j \mid j \in [1, \dots, K]$ be the selected action of $\mathrm{LA}_i$.
Let $Z = \{Z_1, \dots, Z_q\}$ denotes a subset of action set $\alpha$.
Let $Z = \{\alpha_1, \dots, \alpha_r\} - \alpha_j \mid Z \subset \alpha$ be the subset of $\mathrm{LA}_i$ action set except for $\alpha_j$, where $q = r - 1$.
begin
  if $signal = 0$
    Reward $\alpha_j$ action of $\mathrm{LA}_i$.
    Penalize $Z$ action members of $\mathrm{LA}_i$: $Z_l \mid l \in [1, \dots, q]$.
    Update corresponding probability vector of $\mathrm{LA}_i$ by using (3).
  else if $signal = 1$
    Penalize $\alpha_j$ action of $\mathrm{LA}_i$.
    Reward $Z$ action members of $\mathrm{LA}_i$: $Z_l \mid l \in [1, \dots, q]$.
    Update corresponding probability vectors of $\mathrm{LA}_i$ by using (4).
  end if
end
```

Ensemble update Regardless of the number of swarm members, all the swarm components are evaluated together in the synergy step and the reinforcement signal is calculated based on either *gbest* information at population level or *sbest* information at the swarm level. The signal deploys only on corresponding learning automata which are associated with the specified swarm. If at least one of the swarm members can improve the *gbest* fitness, this swarm might have a good configuration of dimensions. Such configuration is expected to include more correlated variables of the search space and the associated learning automata may identify the coherent choice of dimensions. Procedure 6 illustrates the grouping update procedure in which the set of learning automata associated with jth swarm retrieves the reward or penalty signal.

Action selection At the end of each generation, the swarm table should be refilled based on the new probability vectors of the automata. The updating procedure allows the algorithm to detect the correlated dimensions more accurately. Thus, at the end of each generation, each automaton determines for every attached dimension which swarm it belongs to (Procedure 7).

The intelligent framework of ACPSO algorithm employs a set of learning automata and utilizes the correlated variables to extend the global and local search abilities of the cooperative PSO. The pseudocode of the ACPSO algorithm is presented in Algorithm 4.

5 Simulation results

In this section, the benchmark functions which are used for experimental simulation and modeling are briefly introduced. In order to examine the performance of the proposed algorithm, four kinds of experiments are conducted in the following subsections. The first experiment is conducted on TEC 2006 [18] benchmark functions which are proposed in [18] to test the CLPSO algorithm. The second experiment consists of optimization problems of TEC 2006 [18] which are unrotated and rotated functions. This set of new benchmarks shows how interestingly ACPSO optimizes the rotated problems and suppresses its competitors. The third selected set of test functions is the composition benchmark functions of CEC 2005 [17], which are quite difficult optimization problems. The fourth and last experiment is carried out on six 300-dimensional multimodal functions of TEC 2009 [20] with the aim of extending the ACPSO application to real world optimization problems.

Since the aim of the ACPSO algorithm is to learn the correlated variables of the search space, the learning parameters and swarm size play an important role in achieving better results. The number of generations which is considered for the symbiosis step is set to 500. Due to this restriction, the associated set of learning automata should learn the correlated variables of the search space in the symbiosis step much faster. Moreover, convergence to the optimal action will occur later in the synergy step.

The ACPSO algorithm was tested on various settings of parameters and finally the $L_{R\text{-}P}$ learning algorithm was selected with learning parameters $alpha = beta = 0.1$. The number of swarms in the ACPSO algorithm is a static parameter which should be set prior to running the algorithm. In order to observe the impact of this parameter, it is set to 3 and 6 for all 30 dimensional experiments conducted in this

Procedure 6 Ensemble update (*signal*, j)

```
define
Reinforcement signal $signal$, current swarm $j$, current dimension $i$, number of swarms $K$ and number of dimensions $N$.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms.
Let $D = \{D_1, \dots, D_N\}$ denotes the set of problem dimensions.
Let $\mathrm{LA} = \{\mathrm{LA}_1, \dots, \mathrm{LA}_N\}$ denotes the set of learning automata designated to each dimension.
Find $S_j$ swarm members:
  Let $d \subseteq D \mid \mathrm{SWARM\_TABLE}(j, d) = 1$ denotes a subset of dimensions set $D$.
  Let $d = \{d_1, \dots, d_p\}$ be the set of corresponding dimensions of $S_j$, where $|d| = p$.
Find set of learning automata associated to $S_j$ for updating their probability vectors by input signal.
  Let $\mathrm{LA\_}S = \{\mathrm{LA\_}S_1, \dots, \mathrm{LA\_}S_p\}$ donates a subset of LA corresponding to $S_j$.
  Let $\mathrm{LA\_}S = \{\mathrm{LA}_1, \dots, \mathrm{LA}_N\} - \{\mathrm{LA}_{d1}, \dots, \mathrm{LA}_{dp}\} \mid \mathrm{LA\_}S \subseteq \mathrm{LA}$ be the of corresponding learning automata of $S_j$.
Let $\alpha = \{\alpha_1, \dots, \alpha_r\}$ denotes the action set of each automaton of LA_$S$, where $r = K$.
Let $\alpha_j \mid j \in [1, \dots, K]$ denotes the selected action of LA_$S$ which are designated to $S_j$.
Let $Z = \{Z_1, \dots, Z_q\}$ denotes a subset of action set $\alpha$.
Let $Z = \{\alpha_1, \dots, \alpha_r\} - \alpha_j \mid Z \subset \alpha$ be the subset of $\mathrm{LA\_}S_d \mid d = \{d_1, \dots, d_q\}$ action set except for $\alpha_j$, where $q = r - 1$.
begin
  for each $\mathrm{LA\_}S_i \mid i \in [1, \dots, p]$:
    if $signal = 0$
      Reward $\alpha_j$ action of $\mathrm{LA\_}S_i$.
      Penalize $Z$ action members of $\mathrm{LA\_}S_i$: $Z_l \mid l \in [1, \dots, q]$.
      Update corresponding probability vector of $\mathrm{LA\_}S_i$ by using (3).
    else if $signal = 1$
      Penalty $\alpha_j$ action of $\mathrm{LA\_}S_i$.
      Reward $Z$ action members of $\mathrm{LA\_}S_i$: $Z_l \mid l \in [1, \dots, q]$
      Update corresponding probability vectors of $LA\_S_i$ by using (4).
    end if
end for
end
```

Procedure 7 Action selection

```
define
Number of swarms $K$ and number of dimensions $N$.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms disseminated in problem dimensions.
Let $\mathrm{LA} = \{\mathrm{LA}_1, \dots, \mathrm{LA}_N\}$ denotes the set of learning automata designated to each dimension.
Let $\alpha = \{\alpha_1, \dots, \alpha_r\}$ denotes the action set of each automaton, where $r = K$.
Let $p = \{p_1, \dots, p_r\}$ denotes the probability vector corresponding to action set of each automaton.
begin
  for each $\mathrm{LA}_i \mid i \in [1, \dots, N]$
    $\mathrm{LA}_i$ selects the $\alpha_j$ action based on its current probability vector $p$ from its action set $\alpha$.
    Let $\alpha_j \mid j \in [1, \dots, K]$ be the selected action of $\mathrm{LA}_i$.
    Let $Z = \{Z_1, \dots, Z_q\}$ denotes a subset of action set $\alpha$.
    Let $Z = \{\alpha_1, \dots, \alpha_r\} - \alpha_j \mid Z \subset \alpha$ be the subset of $\mathrm{LA}_i$ action set except for $\alpha_j$, while $q = r - 1$.
    Update swarm table:
      Fill $j$th row of swarm table with 1
        $\mathrm{SWARM\_TABLE}(j, i) = 1$
      Fill $Z$ row members of swarm table with 0:
        $\mathrm{SWARM\_TABLE}(Z_l, d) = 0 \mid l \in [1, \dots, q]$
  end for
end
```

Algorithm 4 ACPSO algorithm

```
define
Initialize PSO parameters: population size $PS$, dimension number $N$, number of swarms $K$, generation $ge = 0$, fitness
evaluation $fe = 0$, maximum fitness evaluations $FE$, maximum generations $GE$, train epoch $TE$, Improvement Tag $t = 0$
and inertia weight $w$.
Initialize SWARM_TABLE$_{[K\times N]}$ data structure.
Initialize position $x$ and associated velocity $v$.
Initialize $pbest = [pbest_1, \ldots, pbest_N]$ and $gbest = [gbest_1, \ldots, gbest_N]$ of population.
Initialize $K$ swarms: $S = \{S_1, \ldots, S_K\}$
Initialize $sbest_j \mid j \in [1, \ldots, K]$
Let $D = \{D_1, \ldots, D_N\}$ be the set of problem dimensions.
Let $A(d) \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in the corresponding $d$ positions of vector $A$. Also,
let $A \leftarrow B(d) \mid d \subseteq D$ overwrites $d$ positions of vector $B$ in vector $A$.
Initialize LA parameters: action probability vector $P$, alpha (reward signal), beta (penalty signal) and action number $r$,
where $r = K$.
Call procedure ACTION_SELECTION and fill SWARM_TABLE.
Call procedure INITIALIZE_CONTEXT_VECTOR.
Let $\mathbf{b}(d, p)$ as procedure CONTEXT_VECTOR (dimension, position) which builds context vector in different contexts of
different swarms.
Let $f$ be the fitness value.
repeat      // Symbiosis phase
  for each swarm $j \in [1, \ldots, K]$:
    find $S_j$ swarm members:
      Let $d$ be the set of $S_j$ dimension members.
      $d \subseteq D|$ SWARM_TABLE$(j, d) = 1$
    for each particle $i \in [1, \ldots, PS]$:
      Evaluate the particle through the context vector:
        Let $p$ be the position that is going to evaluate in the context of $S_j$.
        Call procedure $\mathbf{b}(d, p)$ for evaluating $x_i$ particle of $S_j$ swarm in the context of $S_j$.
        if $f(\mathbf{b}(d, x_i(d)) < f(\mathbf{b}(d, pbest_i(d)))$ then
          $pbest_i(d) \leftarrow x_i(d)$
        end if
        if $f(\mathbf{b}(d, pbest_i(d))) < f(\mathbf{b}(d, gbest(d)))$ then
          $gbest(d) \leftarrow pbest_i(d)$
          $sbest_j \leftarrow pbest_i(d)$
          $t = 1$
        end if
      $fe = fe + 1$
    end for
    Perform PSO update for Velocity and Position on $S_j$ using (1) and (2).
    if $g > 0$ AND $j == (g \bmod N) + 1$
      Call procedure REINFORCEMENT_SIGNAL $(t, j)$      // calculate signal
      Call procedure ISOLATE_UPDATE (signal,$j$)      // deploy signal
    end if
    $t = 0$
  end for
  Call procedure ACTION_REFINEMENT $(g)$ and refine the action of associated LA.
  $ge = ge + 1$
until $ge < EP$
repeat      // Synergy phase
  for each swarm $j \in [1, \ldots, K]$:
    find $k_j$ swarm members:
      $d \subseteq D \mid$ SWARM_TABLE$(j, d) = 1$
```

Algorithm 4 (*Continued*)

```
       Let $d$ be the set of $S_j$ dimension members.
    for each particle $i \in [1, \ldots, PS]$:
      Evaluate the particle through the context vector:
        Let $p$ be the position that is going to evaluate in the context of $S_j$.
        Call procedure $\mathbf{b}(d, p)$ for evaluating $x_i$ particle of $S_j$ swarm in the context of $S_j$.
        if $f(\mathbf{b}(d, x_i(d)) < f(\mathbf{b}(d, pbest_i(d)))$ then
          $pbest_i(d) \leftarrow x_i(d)$
        end if
        if $f(\mathbf{b}(d, pbest_i(d))) < f(\mathbf{b}(d, gbest(d)))$ then
          $gbest(d) \leftarrow pbest_i(d)$
          $sbest_j \leftarrow pbest_i(d)$
          $t = 1$
        end if
      $fe = fe + 1$
    end for
      Perform PSO update for Velocity and Position on $S_j$ using (1) and (2).
      Call procedure REINFORCEMENT_SIGNAL $(t, j)$.      // calculate signal
      Call procedure ENSEMBLE_UPDATE (signal,$j$).      // deploy signal
      $t = 0$
  end for
  Call procedure ACTION_SELECTION and fill SWARM_TABLE for the next generation.
  $ge = ge + 1$
until ($ge <= G$ AND $fe <= FE$)
```

paper and also 30 and 60 for 300-dimensional experiment. Tables 1–3 summarize the experimental results for 30D problems while Table 4 lists those from 300-dimensional cases.

5.1 Experiment 1: IEEE TEC 2006 benchmark functions

5.1.1 *Test functions and parameter settings*

There are 16 different optimization benchmark functions which are introduced in TEC 2006 [18]. These test functions are categorized in four main groups:

Group A: Unimodal and Simple Multimodal Problems: f_1–f_2
Group B: Unrotated Multimodal Problems: f_3–f_8
Group C: Rotated Multimodal Problems: f_9–f_{14}
Group D: Composition Problems: f_{15}–f_{16}

To show the performance of the proposed method, the ACPSO algorithm was run on these 16 test problems with 30 dimensions and compared with two famous PSO variants. Essentially one of our counterparts is CPSO-H [8] and the other one is CLPSO [18]. The parameter settings which are used for these two PSOs are similar to TEC 2006 [18]. For 30-dimensional problems, the population size is set to 40 and the maximum fitness evaluation is set to 200,000, respectively. For each of these test functions, 30 independent runs are conducted and the average and standard deviation of the results being reported in Table 1.

5.1.2 *30-Dimensional problems*

Table 1 shows the overall performance of ACPSO and other algorithms on 16 different 30-dimensional benchmark functions. The obtained results demonstrate that ACPSO alleviates *curse of dimensionality* and retains its performance in 30-D problems. In both unimodal and multimodal problems (f_1–f_8), ACPSO outperforms CPSO-H. The f_1 problem is a simple and convex one which can be optimized by ACPSO faster than other PSOs. In f_2, ACPSO can avoid getting trapped in local minima by changing its search direction through the problem space. In unrotated multimodal problems (f_3–f_8), global and local searches are managed simultaneously. The algorithms that work on dimensions of the problem independently fail to optimize these kinds of problems.

Cooperative algorithms divide the solution space into several subspaces and assign each subspace to a swarm. The cooperation avoids early stagnation of the algorithm and balances both exploration and exploitation features of PSO. ACPSO algorithm generally performed similar to CLPSO and better than CPSO-H in rotated multimodal problems (f_9–f_{14}) and composition functions (f_{15}–f_{16}), which are the most complex problems of this experiment. Since the

Table 1 The mean and standard deviation of function error values for ACPSO-$L_{R\text{-}P}$, CPSO-H [8] and CLPSO [18] algorithms in 30-D problems. The last three rows represent the performance compassion between ACPSO and other algorithms, where "−", "+" and "=" indicate that the performance of ACPSO is worse, better and similar to counterpart algorithms, respectively. The values listed in the "R" columns are used to specify this performance measure

F	CLPSO	R	CPSO-H	R	ACPSO$_3$-$L_{R\text{-}P}$	ACPSO$_6$-$L_{R\text{-}P}$
f_1	4.46E−14 ± 1.73E−14	+	1.16E−113 ± 2.92E−113	+	0.00E+00 ± 0.00E+00	2.43E−188 ± 0.00E+00
f_2	2.10E+01 ± 2.98E+00	+	7.08E+00 ± 8.01E+00	+	3.18E−05 ± 4.01E−05	2.89E−06 ± 2.34E−06
f_3	0.00E+00 ± 0.00E+00	−	4.93E−14 ± 9.17E−14	+	8.76E−15 ± 3.06E−15	1.85E−14 ± 6.00E−15
f_4	3.14E−10 ± 4.64E−10	−	3.63E−02 ± 3.60E−02	+	3.47E−02 ± 2.29E−02	4.84E−02 ± 3.04E−02
f_5	3.45E−07 ± 1.94E−07	+	7.82E−15 ± 8.50E−15	+	0.00E+00 ± 0.00E+00	5.45E−15 ± 5.81E−15
f_6	4.85E−10 ± 3.63E−10	+	0.00E+00 ± 0.00E+00	+	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_7	4.36E−10 ± 2.44E−10	+	1.00E−01 ± 3.16E−01	+	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_8	1.27E−12 ± 8.79E−13	−	1.83E+03 ± 2.59E+02	+	8.94E+02 ± 8.14E+01	4.93E+02 ± 1.22E+02
f_9	3.43E−04 ± 1.91E−04	+	2.10E+00 ± 3.84E−01	+	8.20E−15 ± 2.61E−15	1.65E−14 ± 4.65E−15
f_{10}	7.04E−10 ± 1.25E−11	−	5.54E−02 ± 3.97E−02	+	2.85E−02 ± 2.41E−02	3.61E−02 ± 3.48E−02
f_{11}	3.07E+00 ± 1.61E+00	−	1.43E+01 ± 3.53E+00	+	2.95E+00 ± 1.33E+00	3.62E+00 ± 1.41E+00
f_{12}	3.46E+01 ± 1.61E+00	+	1.01E+02 ± 3.53E+00	+	3.27E+01 ± 5.07E+00	3.61E+01 ± 8.13E+00
f_{13}	3.77E+01 ± 5.56E+00	+	8.80E+01 ± 2.59E+01	+	2.58E+01 ± 6.29E+00	2.95E+01 ± 4.21E+00
f_{14}	1.70E+03 ± 1.86E+02	−	3.64E+03 ± 7.41E+02	+	3.89E+03 ± 5.72E+02	3.51E+03 ± 4.20E+02
f_{15}	7.50E−05 ± 1.85E−04	+	1.30E+02 ± 1.64E+02	+	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{16}	7.86E+00 ± 3.64E+00	−	7.83E+01 ± 1.60E+02	−	5.27E+02 ± 1.10E+02	4.97E+02 ± 1.17E+02
+		9		15		
−		7		1		
=		1		0		

swarm members are stochastically selected in ACPSO and there is no helpful feedback about the swarm configuration, the results are tolerable as compared to the CLPSO algorithm. Population of CLPSO learns from different exemplars in different dimensions, which implies the large potential search space of CLPSO. For this purpose, the CLPSO algorithm emphasizes diverse feasible solutions of the search space. This set of diverse solutions which originate from the theoretical search experience of each particle, leading to superior performance of CLPSO on highly multimodal problems. Having a look at rank (R) columns of Table 1 reveals that ACPSO statistically outperforms CLPSO and CPSO-H in 62.5 % and 93.75 % of the test cases, respectively.

If the dimensionality of the problem increases, then the optimization of it by PSO will be become more difficult. This is exactly due to exponential growth of the search space. In order to confront the mentioned problem known as *curse of dimensionality*, the volume of the search space is divided and assigned to small swarms in cooperative approaches. ACPSO algorithm dynamically exploits these small volumes during the evolution and modifies their configuration in an attempt to discover correlated components of the search space. So in the ACPSO algorithm, the results will not deteriorate if the dimensionality of the problem increases.

In TEC 2006 benchmark functions [18], increasing the number of swarms from 3 to 6 incurs a detrimental effect on the performance of the ACPSO algorithm. This phenomenon especially occurs in 30-D problems of TEC2006 and almost fades in 300-D problems of [20]. Large numbers of swarms suffer from two major drawbacks: first, it increases the number of function evaluations needed for convergence of the algorithm; and second, it leads to a large action set per automaton. When there are no sub correlated variables in the search space to produce a suitable reinforcement signal, the automata cannot select an optimal policy efficiently. In an attempt to enhance the performance of the ACPSO algorithm, a set of new benchmarks are conducted in the next experiment, which precisely uncovers a subtle performance for ACPSO.

5.2 Experiment 2: active coordinate rotated benchmark functions

To have a fair compassion of ACPSO and other PSOs, one should design some new test functions with correlated dimensions. In order to show the correlation detection feature

Algorithm 5 Pseudocode of active rotation matrix

```
Define
Rotation matrix $M$, number of dimensions $N$, number of swarms $K$ and swarm length $L$.
Let $x = [x_1, \dots, x_D]^T$ be the original variable.
Let $y = [y_1, \dots, y_D]^T$ be the new rotated variable.
Let $S = \{S_1, \dots, S_K\}$ denotes the set of swarms.
Let $D = \{D_1, \dots, D_N\}$ denotes the set of problem dimensions.
$K_1 = n \bmod K$; $L_1 = \lceil \frac{N}{K} \rceil$
$K_2 = K - (n \bmod K)$; $L_2 = \lfloor \frac{N}{K} \rfloor$
for each $K_i \mid i \in [1, 2]$
  Initialize $K_i L_i$-dimensional rotation matrix:
    for each swarm $S_j \mid j \in [1, \dots, K_i]$
      $M_j = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1L_i} \\ m_{21} & m_{22} & \dots & m_{2L_i} \\ \dots & \dots & \dots & \dots \\ m_{L_i1} & m_{L_i2} & \dots & m_{L_iL_i} \end{bmatrix}$
    Split the corresponding dimensions of $S_j$:
      Let $r = [1, \dots, d]$ be the range of associated swarm, where $d = L_i$.
      $y_j = [x_1, \dots, x_d]^{\mathrm{T}} \times M_j$
    end for
end for
$y = [y_1, \dots, y_K]^T$
Use variable $y$ to calculate the fitness value $f$.
```

of ACPSO while solving an optimization problem, some active coordinate rotated benchmark functions are introduced here. Moreover, one should adapt CPSO-H and CLPSO algorithms [8, 18] to fit this new validation method.

5.2.1 *Building new benchmark functions*

A new set of benchmark functions is proposed for correlation detection in this subsection. Originally, these functions are taken from TEC 2006 [18]. Based on the previously mentioned features of ACPSO, capacities of the algorithm emerge entirely just when the coordinate rotation is applied to dimensions of the specific problem. The test functions used in [8, 18], apply the rotation matrix to all dimensions and create a correlation between all dimensions of the problem. Due to the globally coordinate rotated benchmark functions, which are used in [8, 18], the search space is divided into multiple sub dimensions with these sub dimensions being rotated intentionally. This process leads to having some correlated subspaces in the search space.

The pseudocode of the partitioning mechanism is outlined in Algorithm 5 for these new benchmarks. First, the problem dimensions are split and some rotation matrices are created according to the number of swarms. Then, while evaluating each member of the population, these correlated sub dimensions are pre-multiplied to their designated rotation matrix. Finally, multiple correlated sub dimensions are created which are independent from each other. If these correlated variables are put into one same swarm, performance of the algorithm will be improved significantly. This process is applied to coordinate rotated multimodal functions (f_9–f_{14}) of TEC 2006 [18] and leads to the creation of new active coordinate rotated benchmark functions.

Before evaluating the particles by these new benchmark functions, one should adapt PSOs with the proposed procedure. Besides modification of the CLPSO algorithm, two versions of the CPSO-H algorithm are implemented as discussed in the following.

Randomized CPSO-H (rCPSO-H) Due to the nature of the rotation matrix generator, correlated variables of the search space will be placed sequentially together. Since naïve selection of the correlated dimensions in the CPSO-H algorithm is going to be shown, a permutation will be applied to the swarm members at the beginning of this algorithm. Then, the correlated dimensions are not exactly placed in the same swarm. Consequently, this partitioning mechanism has been simulating the hopeless structure of the CPSO-H algorithm in finding some correlated variables for the same swarm.

Idealized CPSO-H (iCPSO-H) Since the partitioning mechanism blindly takes the value c ($c < N$) in CPSO-H [8], some correlated variables are expected to end up in the same swarm. The rotation matrix was exactly applied on sub dimensions of the problem which are placed in the same swarm. As a result, the correlated variables will be optimized together and performance of the algorithm will obviously dominate that of rCPSO-H.

Both rCPSO-H and iCPSO-H algorithms determine their swarm members at the beginning of the algorithm and sustain the initial configuration of the correlated variables during evolution. In the current arranged experiment, the lack of learning ability for CPSO-H will be sensed more considerably and the power for correlation detection of the ACPSO algorithm will emerge strongly.

5.2.2 Benchmark functions and parameter configurations

In the following experiment, the discussed rotation matrix of Algorithm 5 is applied to the multimodal coordinate rotated test functions (f_9–f_{14}) of TEC 2006 [18] and they are employed to compare the proposed method with the CLPSO, rCPSO-H and iCPSO-H algorithms. All the algorithms run on 30-D problems with 40 particles per population. The experiments are repeated 30 times with the standard deviation and mean of these runs being reported in Table 2. Similar to experiment 1, the corresponding set of learning automata use a $L_{R\text{-}P}$ learning algorithm with $alpha = beta = 0.1$.

5.2.3 Discussion of results

All the problems which are used in this experiment are active coordinate rotated multimodal functions. Since swarm members of iCPSO-H are fully tuned on correlated dimensions of the problem space, the performance of iCPSO-H is superior to other algorithms. Although iCPSO-H exactly knows the correlation configuration, ACPSO learning ability can suppresses iCPSO-H in almost half of the benchmark functions. There is a chance that ACPSO produces noisy swarms or roughly finds correlated dimensions. These phenomena may balance the abilities of global and local searches and improve performance of the search to some context.

rCPSO-H algorithm is a cooperative PSO without any information about correlated parameters of the search space (like CPSO-H in [8]). In 30-D problems, ACPSO suppresses rCPSO-H in all test cases almost 100 %. The learning ability of ACPSO is clearly evident in the experiment conducted with rCPSO-H. However, the new learning strategy of CLPSO is based on the *pbest* information from each individual of the population and thus can exploit the required solution diversity from the designed benchmark functions; CLPSO algorithm maintains its performance. By comparing the proposed method with CLPSO, ACPSO outperforms the CLPSO algorithm in 3 out of 6 problems.

From Table 2, one can realize that in addition to simplifying the problem, increasing the number of applied separate rotations will also improve performance of the algorithm. This case obviously occurs in the iCPSO-H algorithm. This algorithm knows the exact location of correlated variables. The ACPSO algorithm tries to find the proper set of dimensions for each of the swarms. This functionality needs a proper feedback from the environment which is defined as the improvement of the *gbest* information of the population. Since the reinforcement signal is defined globally and there are enough generations, the set of learning automata can converge to an optimal configuration of swarms and its performance achieves better results than CLPSO and rCPSO-H.

5.3 Experiment 3: IEEE CEC 2005 benchmark functions

In the special session of CEC 2005 [17], 25 benchmark functions are defined for real-parameter optimization. The mathematical formula and properties of these functions are described in [17]. The test functions are divided into four basic groups:

(1) Unimodal Functions (5)
(2) Basic Multimodal Functions (7)
(3) Expanded Multimodal Functions (2)
(4) Hybrid Composition Functions (11)

To show the eligibility of the ACPSO algorithm in different benchmark functions, in this section ACPSO is compared with the CPSO-H (randomized version) and CLPSO algorithms. For each algorithm, the population size, number of dimensions and termination condition are set to 30 particles, 30 dimensions and 300,000 fitness evaluations (FEs). The results for ACPSO and CPSO-H are reported for 6 swarms per population. All algorithms were run 25 times with the average and standard deviation of function error values ($f(x) - f(x^*)$) being reported in Table 3. Similar to experiment 1 and 2, the learning algorithm is an $L_{R\text{-}P}$ schema with $alpha = beta = 0.1$.

5.3.1 Discussion of results

The experimental results are presented in Table 3. On unimodal functions (f_1–f_5), the CPSO-H algorithm performs very well. Since it partitions the problem space into a fixed number of subspaces and each subpopulation exploits its designated subspace, CPSO-H will outperform other PSOs. Although ACPSO is a cooperative PSO, it doesn't show a steady swarm configuration. As ACPSO may benefit from different parts of the problem space cooperatively, it will not perform the best for unimodal problems. In simple multimodal functions (f_6–f_{12}) ACPSO completely suppresses its parent PSO, i.e. CPSO-H, in 5 out of 7 test functions. Especially in rotated multimodal functions (f_7, f_8, f_{10}, f_{11}) where the problem dimensions are non-separable and correlation occurs between subspaces, ACPSO will perform better than CPSO-H. Each dimension

Table 2 The mean and standard deviation of function error values for ACPSO-$L_{R\text{-}P}$, iCPSO-H, rCPSO-H and CLPSO algorithms in 30-D problems. *The last three columns* represent the performance compassion between ACPSO and other algorithms, where the "−", "+" and "=" indicate that the performance of ACPSO is worse, better and similar to counterpart algorithms, respectively. The values listed in the "R" columns are used to specify this performance measure. The column S represents the number of swarms

Algorithm	S	f_9	R	f_{10}	R	f_{11}	R	+	−	=
CLPSO	3	5.39E−12 ± 2.68E−11	+	3.68E−10 ± 1.84E−09	−	4.77E−01 ± 4.20E−01	−	1	2	0
	6	5.66E−14 ± 1.07E−13		5.47E−15 ± 2.59E−14		1.56E−01 ± 1.48E−01				
iCPSO-H	3	1.65E−01 ± 3.93E−01	+	3.14E−02 ± 3.08E−02	+	1.92E + 00 ± 1.62E+00	−	2	1	0
	6	2.77E−14 ± 6.49E−15		6.35E−02 ± 6.53E−02		6.59E−02 ± 9.52E−02				
rCPSO-H	3	8.53E−01 ± 8.82E−01	+	2.64E−02 ± 2.39E−02	+	6.04E+00 ± 2.09E+00	+	3	0	0
	6	5.76E−01 ± 7.44E−01		3.82E−02 ± 4.38E−02		5.60E+00 ± 2.48E+00				
ACPSO-$L_{R\text{-}P}$	3	7.84E−15 ± 2.20E−15		2.42E−02 ± 2.14E−02		1.74E+00 ± 1.18E+00				
	6	1.29E−14 ± 3.43E−15		4.05E−02 ± 3.78E−02		2.92E+00 ± 1.44E+00				
Algorithm	S	f_{12}	R	f_{13}	R	f_{14}	R	+	−	=
CLPSO	3	2.56E+01 ± 7.62E+00	−	1.99E+01 ± 5.49E+00	+	3.39E+03 ± 4.05E+02	+	2	1	0
	6	2.35E+01 ± 6.38E+00		1.69E+01 ± 3.27E+00		3.11E+03 ± 5.91E+02				
iCPSO-H	3	3.06E+01 ± 7.71E+00	−	1.92E+01 ± 3.90E+00	−	2.77E+03 ± 4.16E+02	−	0	3	0
	6	1.30E+01 ± 3.59E+00		9.40E+00 ± 1.53E+00		1.65E+03 ± 3.51E+02				
rCPSO-H	3	5.11E+01 ± 1.71E+01	+	4.46E+01 ± 1.47E+01	+	3.80E+03 ± 6.20E+02	+	3	0	0
	6	5.06E+01 ± 1.48E+01		2.98E+01 ± 1.45E+01		3.93E+03 ± 5.09E+02				
ACPSO-$L_{R\text{-}P}$	3	2.80E+01 ± 3.92E+00		1.85E+01 ± 2.17E+00		3.60E+03 ± 4.14E+02				
	6	3.14E+01 ± 7.30E+00		1.59E+01 ± 2.81E+00		3.06E+03 ± 4.43E+02				

learns from different exemplars in CLPSO, so it achieves superior results over ACPSO in unrotated multimodal functions (f_6, f_9).

The expanded multimodal functions (f_{13}–f_{14}) and hybrid decomposition functions (f_{15}–f_{25}) are the hardest ones to optimize since they mix properties of different functions together. The intelligence dimension selection of ACPSO leads to an efficient sub dimension configuration, thus it reduces the effectiveness of decomposition of the problems. The aim of learning automata is to exploit the correlation by using multiple swarms and maintaining a mechanism simultaneously for these swarms in order to cooperate toward solving the problem. The performance of ACPSO is better than CLPSO and CPSO-H on seven and thirteen test functions, respectively and is similar to CLPSO in two problems.

In summary, the adaptive cooperative approach offered performance improvement in terms of correlation detection. In unimodal and simple multimodal functions (groups A and B) ACPSO performs much poorer than CPSO-H. The dimensions of these test cases are separable and they could be easily optimized by splitting them into some fixed subswarms. Since ACPSO couldn't find any proper correlation in the search space, the swarm configuration of it might vary during the evolution and achieves rather weak results in comparison with CPSO-H. Although in hybrid composition test functions [17, 65] there is no active coordinate rotation test function like test cases of experiment 2, ten different functions are dealt with simultaneously here. Meanwhile, a flexible mechanism is still needed to balance exploration and exploitation in these tough problems. The column R in Table 3 indicates that, ACPSO has collectively performed better than the CLPSO and CPSO-H algorithms in 68 % and 76 % of test functions, respectively.

5.4 Experiment 4: high dimensional multimodal benchmark functions

Real world optimization problems probably contain more than hundreds of variables. The GSO introduced in [20], [21] is used to handle large-scale optimization problems. In [20], GSO is tested on six 300-D multimodal test functions (f_8–f_{13}). In order to evaluate ACPSO performance on high dimensional problems, these benchmark functions are used. The results of ACPSO are compared with CPSO-H [8], CLPSO [18] and GSO [20]. Similar to the experimental setting of [20], population size and maximum number of function evaluations are set to 50 and 3,750,000, respectively. Note that according to [20] the population size of GSO is set to 48. All experiments were run for 5 trials and the reported results are the final average from the four algorithms. In this

Table 3 The mean and standard deviation of function error values for ACPSO-L_{R-P}, CPSO-H and CLPSO [65] algorithms for 30-D problems. The *last three rows* represent the performance compassion between ACPSO and other algorithms, where the "−", "+" and "=" indicate that the performance of ACPSO is worse, better and similar to counterpart algorithms, respectively. The values listed in the "R" columns are used to specify this performance measure

F		CLPSO	R	CPSO-H	R	ACPSO-L_{R-P}
Unimodal functions	f_1	0.00E+00 ± 0.00E+00	−	1.32E−13 ± 3.16E−14	−	2.08E−03 ± 7.21E−04
	f_2	8.40E+02 ± 1.90E+02	+	3.19E−02 ± 5.14E−02	−	3.33E+00 ± 1.59E+00
	f_3	1.42E+07 ± 4.19E+06	+	5.02E+05 ± 2.77E+05	−	4.03E+06 ± 1.48E+06
	f_4	6.99E+03 ± 1.73E+03	+	7.72E−02 ± 1.38E−01	−	2.89E+00 ± 1.03E+00
	f_5	3.86E+03 ± 4.35E+02	+	6.88E+03 ± 2.20E+03	+	5.52E+03 ± 1.04E+03
Basic multimodal functions	f_6	4.16E+00 ± 3.48E+00	−	7.11E+01 ± 1.53E+02	−	1.01E+02 ± 4.07E+01
	f_7	4.51E−01 ± 8.47E−02	+	2.57E−02 ± 1.89E−02	+	1.72E−01 ± 5.78E−02
	f_8	2.09E+01 ± 4.41E−02	+	2.03E+01 ± 7.21E−02	+	2.00E+01 ± 9.40E−03
	f_9	0.00E+00 ± 0.00E+00	−	9.95E−01 ± 9.53E−01	+	1.16E−02 ± 5.36E−03
	f_{10}	1.04E+02 ± 1.53E+01	−	1.90E+02 ± 6.32E+01	+	1.25E+02 ± 2.12E+01
	f_{11}	2.60E+01 ± 1.63E+00	+	2.66E+01 ± 3.56E+00	+	2.06E+01 ± 3.22E+00
	f_{12}	1.79E+04 ± 5.24E+03	+	2.64E+03 ± 4.38E+03	−	1.60E+04 ± 5.57E+03
Expanded multimodal functions	f_{13}	2.06E+00 ± 2.15E−01	+	1.12E+00 ± 4.70E−01	+	5.77E−01 ± 2.50E−01
	f_{14}	1.28E+01 ± 2.48E−01	+	1.29E+01 ± 5.04E−01	+	1.27E+01 ± 4.08E−01
Hybrid composition functions	f_{15}	5.77E+01 ± 2.76E+01	−	3.57E+02 ± 2.18E+02	+	2.58E+02 ± 7.15E+01
	f_{16}	1.74E+02 ± 2.82E+01	−	3.28E+02 ± 1.49E+02	+	2.20E+02 ± 6.09E+01
	f_{17}	2.46E+02 ± 4.81E+01	+	3.13E+02 ± 1.61E+02	+	1.96E+02 ± 5.52E+01
	f_{18}	9.13E+02 ± 1.42E+00	+	8.34E+02 ± 2.47E+00	+	8.32E+02 ± 1.77E+00
	f_{19}	9.14E+02 ± 1.45E+00	+	8.35E+02 ± 3.39E+00	+	8.33E+02 ± 2.42E+00
	f_{20}	9.14E+02 ± 3.62E+00	+	8.35E+02 ± 3.24E+00	+	8.33E+02 ± 1.74E+00
	f_{21}	5.00E+02 ± 3.39E−13	=	6.06E+02 ± 2.48E+02	+	5.00E+02 ± 2.55E−04
	f_{22}	9.72E+02 ± 1.20E+01	+	8.18E+02 ± 1.66E+02	+	6.42E+02 ± 1.54E+02
	f_{23}	5.34E+02 ± 2.19E−04	=	8.51E+02 ± 2.32E+02	+	5.34E+02 ± 5.06E−04
	f_{24}	2.00E+02 ± 1.49E−12	−	2.92E+02 ± 2.90E+02	+	2.08E+02 ± 5.32E+00
	f_{25}	2.00E+02 ± 1.96E+00	−	2.92E+02 ± 2.90E+02	+	2.08E+02 ± 5.32E+00
	+		15		19	
	−		8		6	
	=		2		0	

experiment, the swarm size is also scaled based on the problem dimensions. Since dimensions are 10 times larger than 30-D problems, the swarm size is set to 60 for each Cooperative PSO. Finally, each automaton of ACPSO uses the $L_{R\text{-}P}$ learning algorithm with $alpha = beta = 0.1$.

Comparing to GSO, CLPSO and CPSO-H, ACPSO performed significantly better on most benchmark functions. The optimal solution for f_9 (Schwefel function) is supposed to be equal to −125351.2 and ACPSO algorithm consistently achieves this optimal value. Meanwhile, this function is quite easy to optimize for both ACPSO and CPSO-H because both algorithms can work on a sub-dimension of the problem which would can increase diversity of the solution greatly.

f_{10} (Ackley's function) is a non-separable benchmark function with numerous local minima. In cooperative frameworks, CPSO-H performed significantly better than ACPSO. The learning automata of ACPSO have 60 actions corresponding to the number of swarms. As an environment, f_{10} test function could not supply a suitable reinforcement signal for learning automata, because the probability vectors of action sets vary during evolution and the selected action out of this large action set needs further steps for reaching convergence. Since ACPSO could not find any correlated dimensions, it wandered around different configuration of the swarm table. But, CPSO-H swarm members are fixed and they exploit specific configurations of dimensions from the beginning of the evolution process.

Table 4 The mean of function error values for ACPSO-$L_{R\text{-}P}$, CPSO-H, CLPSO and GSO [20], algorithms in 300-D problems. The *last three rows* represent the performance comparison between ACPSO and other algorithms, where the "−", "+" and "=" indicate that the performance of ACPSO is worse, better and similar to counterpart algorithms, respectively. The values listed in the "R" columns are used to specify this performance measure

F	GSO	R	CLSPO	R	CPSO-H	R	ACPSO-$L_{R\text{-}P}$
f_8	−125351.2	+	−12566.7	+	−12569.4	+	−12569.5
f_9	9.89E+02	+	5.87E+01	+	4.03E−01	+	1.02E−07
f_{10}	1.35E−03	+	8.36E−01	+	8.94E−08	−	1.01E−04
f_{11}	1.82E−07	+	4.91E−02	+	4.08E−02	+	6.06E−09
f_{12}	8.26E−08	+	2.28E−02	+	1.49E−07	+	1.85E−10
f_{13}	2.02E−07	+	8.86E−02	+	1.68E−05	+	2.63E−08
+		6		6		5	
−		0		0		1	
=		0		0		0	

f_{11} (Griewank's function) has a cosine term which makes their problem more difficult to optimize. An interesting phenomenon of this function is that it is more difficult to optimize for lower dimensions (e.g. 30-D problems of experiment 1) than higher dimensions (e.g. 300-D problems of this experiment).

In f_{12} and f_{13} (two generalized penalized functions), ACPSO gets the best results. Furthermore, GSO optimizes these benchmarks properly. There is a random sampling procedure in the GSO algorithm for finding the best fitness value (scrounging) combined with ranging behavior of low number of members. In each searching bout (generation), GSO performs a global optimizer as well as a local optimizer.

From Table 4 it can also be seen that CLPSO cannot be properly scaled to handle 300-D problems despite the satisfactory results of Table 1 (30-D problems). Finally, ACPSO outperformed the GSO, CLPSO, and CPSO-H algorithms in 100 %, 100 %, and 83 % of test functions.

6 Conclusion

Many PSOs have been introduced in the literature for global numerical optimization but none of them aims to intelligently exploit from specific sub dimensions of the search space. The ACPSO proposed in this paper followed an adaptive scheme along this goal. It used cooperative swarms to optimize the problem and also employed a learning automaton on each dimension to define membership of the swarms. In ACPSO, each automaton generated its own action based on its probability vector and assigned the corresponding dimension into a swarm.

Four different kinds of experiments were carried out in this paper including three state-of-the-art function optimization benchmark functions in addition to one new set of benchmark functions. In 30-D problems, ACPSO was compared with two other well-known PSO versions, i.e. CLPSO and CPSO-H. Since the correlation detection of ACPSO needs an efficient feedback from the problem, it couldn't perform its best on unimodal and simple multimodal. This behavior is due to the separable search space of these test functions which were addressed by designing several active coordinate rotated benchmark functions. The ACPSO utilizes the increase in the degree of correlation between the subspaces, and performs very well on rotated and composition cases. ACPSO was also used as a high-dimensional optimizer and was compared with three other EAs. The results show that ACPSO is very effective in tackling 300 dimensional problems.

Obviously, the "no free lunch" theorem [66] has emerged in the searching behavior of ACPSO. Although ACPSO is not very successful in solving simple problems, the environment structure is unknown from the beginning when solving real-life problems. Thus, having an adaptive learning mechanism could be really beneficial in solving highly complicated and large-scale problems. Simple problems will be optimized more efficiently by combining ACPSO with a PSO technique such as standard PSO or Comprehensive Learning PSO (CLPSO).

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

References

1. Kennedy J (2006) Swarm intelligence. In: Handbook of nature-inspired and innovative computing, pp 187–219
2. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings IEEE international conference on neural networks, 1995. vol 4, pp 1942–1948

3. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, MHS'95, pp 39–43
4. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: IEEE swarm intelligence symposium, 2007, SIS 2007, pp 120–127
5. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–72
6. Potter M, De Jong K (1994) A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from nature—PPSN III, pp 249–257
7. van den Bergh F, Engelbrecht AP (2000) Cooperative learning in neural networks using particle swarm optimizers. South Afr Comput J 26:84–90
8. van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
9. Narendra KS, Thathachar M (1974) Learning automata: a survey. IEEE Trans Syst Man Cybern 4:323–334
10. Narendra KS, Thathachar MAL (1989) Learning automata: an introduction. Prentice Hall, New York
11. Ünsal C (1997) Intelligent navigation of autonomous vehicles in an automated highway system: Learning methods and interacting vehicles approach, Virginia Polytechnic Institute and State University
12. Beigy H, Meybodi MR (2010) Cellular learning automata with multiple learning automata in each cell and its applications. IEEE Trans Syst Man Cybern, Part B, Cybern 40(1):54–65
13. Esnaashari M, Meybodi MR (2010) Dynamic point coverage problem in wireless sensor networks: a cellular learning automata approach. J Ad Hoc Sens Wirel Netw 10(2–3):193–234
14. Hashemi AB, Meybodi MR (2011) A note on the learning automata based algorithms for adaptive parameter selection in PSO. Appl Soft Comput 11(1):689–705
15. Hashemi A, Meybodi M (2009) Cellular PSO: a PSO for dynamic environments. In: Advances in computation and intelligence, pp 422–433
16. Thathachar M, Sastry PS (2002) Varieties of learning automata: an overview. IEEE Trans Syst Man Cybern, Part B, Cybern 32(6):711–722
17. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technol. Univ., Singapore, IIT Kanpur, Kanpur, India, #2005005, May 2005
18. Liang J, Qin A, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295
19. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
20. He S, Wu Q, Saunders J (2009) Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 13(5):973–990
21. He S, Wu Q, Saunders J (2006) A novel group search optimizer inspired by animal behavioural ecology. In: IEEE Congress on evolutionary computation. CEC 2006, pp 1272–1278
22. Bellman R (1956) Dynamic programming and Lagrange multipliers. Proc Natl Acad Sci USA 42(10):767
23. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. An overview. Swarm Intell 1(1):33–57
24. Lim A, Lin J, Xiao F (2007) Particle swarm optimization and hill climbing for the bandwidth minimization problem. Appl Intell 26(3):175–182
25. Khan SA, Engelbrecht AP (2012) A fuzzy particle swarm optimization algorithm for computer communication network topology design. Appl Intell 36(1):161–177
26. Chu CP, Chang YC, Tsai CC (2011) PC 2 PSO: personalized e-course composition based on particle swarm optimization. Appl Intell 34(1):141–154
27. Wang K, Zheng YJ (2012) A new particle swarm optimization algorithm for fuzzy optimization of armored vehicle scheme design. Appl Intell 37(4):520–526
28. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: The 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence, 1998, pp 69–73
29. Zhan ZH, Zhang J, Li Y, Chung HSH (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern, Part B, Cybern 39(6):1362–1381
30. Zhu Z, Zhou J, Ji Z, Shi YH (2011) DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm. IEEE Trans Evol Comput 15(5):643–658
31. Ji, Z, Liao H, Wang Y, Wu QH (2007) A novel intelligent particle optimizer for global optimization of multimodal functions. In: IEEE congress on evolutionary computation. CEC 2007, pp 3272–3275.
32. Zhan Z-H, Zhang J, Li Y, Shi Y-H (2011) Orthogonal learning particle swarm optimization. IEEE Trans Evol Comput 15(6):832–847
33. Zhang Q, Leung Y-W (1999) An orthogonal genetic algorithm for multimedia multicast routing. IEEE Trans Evol Comput 3(1):53–62
34. Norouzzadeh MS, Ahmadzadeh MR, Palhang M (2012) LADPSO: using fuzzy logic to conduct PSO algorithm. Appl Intell 37(2):290–304
35. Blackwell T, Branke J (2006) Multiswarms, exclusion, and anticonvergence in dynamic environments. IEEE Trans Evol Comput 10(4):459–472
36. Niu B, Zhu Y, He X, Wu H (2007) MCPSO: a multi-swarm cooperative particle swarm optimizer. Appl Math Comput 185(2):1050–1062
37. Zhang J, Ding X (2011) A multi-swarm self-adaptive and cooperative particle swarm optimization. Engineering applications of artificial intelligence
38. Ali YMB (2012) Psychological model of particle swarm optimization based multiple emotions. Appll Intell 36(3):649–663
39. Ong YS, Keane AJ, Nair PB (2002) Surrogate-assisted coevolutionary search. In: Proceedings of the 9th international conference on neural information processing, ICONIP'02, vol 3, pp 1140–1145
40. Bäck T, Schwefel HP (1993) An overview of evolutionary algorithms for parameter optimization. Evol Comput 1(1):1–23
41. Sofge D, De Jong K, Schultz A (2002) A blended population approach to cooperative coevolution for decomposition of complex problems. In: Proceedings of the 2002 Congress on evolutionary computation. CEC'02, vol 1, pp 413–418
42. Han MF, Liao SH, Chang JY, Lin CT (2012) Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl Intell. doi:10.1007/s10489-012-0393-5
43. Shi Y, Teng H, Li Z (2005) Cooperative co-evolutionary differential evolution for function optimization. In: Advances in natural computation, p 428
44. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
45. Cuevas E, Sención F, Zaldivar D, Pérez-Cisneros M, Sossa H (2012) A multi-threshold segmentation approach based on Artificial Bee Colony optimization. Appl Intell 37(3):321–336
46. El-Abd M (2010) A cooperative approach to the artificial bee colony algorithm. In: 2010 IEEE congress on evolutionary computation (CEC), pp 1–5

47. Esnaashari M, Meybodi MR (2011) A cellular learning automata-based deployment strategy for mobile wireless sensor networks. J Parallel Distrib Comput 71:988–1001
48. Akbari Torkestani J, Meybodi MR (2011) A cellular learning automata-based algorithm for solving the vertex coloring problem. Expert Syst Appl 38:9237–9247
49. Misir M, Wauters T, Verbeeck K, Vanden Berghe G (2012) A hyper-heuristic with learning automata for the traveling tournament problem. In: Metaheuristics: intelligent decision making
50. Noroozi V, Hashemi A, Meybodi M (2011) CellularDE: a cellular based differential evolution for dynamic optimization problems. In: Adaptive and natural computing algorithms, pp 340–349
51. Vafashoar R, Meybodi MR, Momeni Azandaryani AH (2011) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell
52. Cheshmehgaz HR, Haron H, Maybodi MR (2011) Cellular-based population to enhance genetic algorithm for assignment problems. Am J Intell Syst 1(1):1–5
53. Wallenta C, Kim J, Bentley PJ, Hailes S (2010) Detecting interest cache poisoning in sensor networks using an artificial immune algorithm. Appl Intell 32(1):1–26
54. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, pp 169–178
55. Farahani SM, Abshouri AA, Nasiri B, Meybodi M (2012) Some hybrid models to improve firefly algorithm performance. Int J Artif Intell 8(S12):97–117
56. Rezvanian A, Meybodi MR (2010) LACAIS: learning automata based cooperative artificial immune system for function optimization. In: 3rd international conference on contemporary computing (IC3 2010), CCIS, 2010, Noida, India. Contemporary computing, vol 94 pp 64–75
57. Meybodi M, Beigy H (2002) A note on learning automata-based schemes for adaptation of BP parameters. Neurocomputing 48(1):957–974
58. Rastegar R, Meybodi MR, Badie K (2004) A new discrete binary particle swarm optimization based on learning automata. In: Proceedings international conference on machine learning and applications, 2004, pp 456–462
59. Jafarpour B, Meybodi M, Shiry S (2007) A hybrid method for optimization (discrete PSO+ CLA). In: International conference on intelligent and advanced systems. ICIAS 2007, pp 55–60. 2007
60. Sheybani M, Meybodi MR (2007) PSO-LA: a new model for optimization. In: Proceedings of 12th annual CSI computer conference of Iran, pp 1162–1169
61. Soleimanzadeh R, Farahani BJ, Fathy M (2010) PSO based deployment algorithms in hybrid sensor networks. Int J Comput Sci Netw Secur 10(7):167–171
62. Hamidi M, Meybodi MR (2008) New learning automata based particle swarm optimization algorithms. In: Iran data mining conference (IDMC), pp 1–15
63. Hasanzadeh M, Meybodi MR, Shiry S (2011) Improving learning automata based particle swarm: an optimization algorithm. In: 12th IEEE international symposium on computational intelligence and informatics, Budapest
64. Hasanzadeh M, Meybodi MR, Ebadzadeh MM (2012) A robust heuristic algorithm for cooperative particle swarm optimizer: a learning automata approach. In: 20th Iranian conference on electrical engineering (ICEE), pp 656–661
65. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
66. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Mohammad Hasanzadeh received his B.Sc. degree in Software Engineering from the Payame Noor University (PNU), Birjand, Iran, in 2009. Currently he is M.Sc. student of Artificial Intelligence in Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. 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 the 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 Paris-Tech, 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.