A Hybrid Method for Optimization (Discrete PSO + CLA)

B. Jafarpour M. R. Meybodi S. Shiry
Computer Engineering and Information Technology Department
Amirkabir University of Technology
Tehran Iran
Email: (jafarpour@cic.aut.ac.ir, mmeybodi@aut.ac.ir, shiry@aut.ac.ir)

***Abstract*-PSO is an evolutionary algorithm that is inspired from collective behavior of animals such as fish schooling or bird flocking. One of the drawbacks of this model is premature convergence and trapping in local optima. In this paper we propose a solution to this problem in discrete version of PSO that uses Learning Automata and introduce a Cellular Learning Automata (CLA) based discrete PSO. Experimental results on five optimization problems show the superiority of the proposed algorithm.**

Keywords: Particle Swarm Optimization, Learning Automata, Cellular Learning Automata, Optimization

I. INTRODUCTION

Particle Swarm Optimization (PSO) method was first proposed by Kennedy and Eberhart [1] in 1995. According to PSO, the behavior of each particle is affected by the best solution that is found by that particle and the best global particle to help it fly through a search space. Moreover, a particle can learn from its past experience to adjust its flying speed and direction. Therefore, by observing the behavior of the flock and memorizing their flying histories all particles in swarm can quickly converge to near optimal geographical with a well preserved population density distribution [20]. PSO is considered as an evolutionary computation approach in that it possesses many characteristics that are used by evolutionary algorithms such as, initializing with a population of random solutions, searching for optima by updating generations, the adjustment of individuals and evaluating them by a fitness function. However unlike evolutionary algorithms, the updates of particles are not accomplished by crossover or mutation[8]. The particle swarm algorithms reported in the literatures are classified into two groups: discrete PSO and continuous PSO [1][2][3]. In continuous PSO the particles operate in continuous search space, where the trajectories are defined as changes in position on some number of dimensions. But in discrete PSO the particles operates on discrete search space, and the trajectories are defined as changes in the probability that a coordinate will take on a value from feasible discrete values. One of the drawbacks of standard PSO model is premature convergence and trapping in local optima. Recently three solutions based on learning automata for solving this problem have been proposed [3][11][12].

In [3] a discrete version of PSO based on learning automata is proposed. In this algorithm, learning automata are used by the particles to model the dynamics of the group to which the particles belong. The set of leaning automata associated to a particle, by observing the behavior of the group, help the particle in searching for optimal geographical with a well preserved population density distribution. In this algorithm the set of learning automata assigned to a particle may be viewed as the brain of the particle determining its position from its own and other particles past experience. To show the effectiveness of the proposed algorithm the authors have tested the algorithm on several function optimization problems. The numerical results have shown that the performance of the proposed algorithm is better than Kennedy's discrete approach for most of the test problems.

In [11] a continuous version of PSO based on learning automata has been proposed. In this model a learning automaton is used to balance exploration and exploitation made by the PSO algorithm. In this paper a new PSO model called PSO-LA is proposed in which a learning automaton takes the role of configuring the behavior of particles and creating a balance between the process of global and local search. The results of experiments conducted by the authors on some standard problems show that the proposed algorithm produces better results than the standard PSO.

In [12] another version of PSO which is a combination of Cellular Learning Automata (CLA)[14] and continuous PSO is proposed. In this model, each cell of CLA contains a population of particles (sub-swarm) and a learning automaton. Learning automaton of each cell is responsible for configuring the behavior of the cell's particles. That is, in each iteration, the learning automaton in each cell of CLA, determines that if the particles should continue their

current way or should follow the best experiences.The act of establishing the balance between global and local search which is done by parameter inertia weight (IW) in [5][19], will be done by learning automata in this model.

In [4][5] neighborhood topology between particles of swarm is considered. In this model the behavior of each particle is affected by the best solution that is found by that particle and the best solution found by the neighboring particles. In this model the best local particle that is selected among topologic neighbors of each particle is replaced by best global particle in standard PSO. In this fashion each particle has its own global best particle. This consideration slows down information flow between particles. This means that a good solution found by a particle needs several generations to be transmitted to other particles while in standard PSO it takes only one generation to happen. This modification prevents crowding of all particles around local optima as it is found by a particle and thus trapping in local optima.

In this paper the Learning Automata based PSO reported in [3] is combined with the neighborhood topology concept and a new discrete PSO based on Cellular Learning Automata is introduced. In the proposed algorithm each cell of CLA contains a particle. Rewards and punishments of learning automata for each particle are determined based on its best location and the best locations of its topologic neighbors. To show the effectiveness of the proposed algorithm we test the algorithm on several function optimization problems. The results of computer simulations show that the proposed algorithm attains better solutions in a faster way for most of the problems.

The rest of the paper is organized as follows. Particle swarm optimization is briefly given in section 2. Section 3 briefly presents Cellular Automata, Learning Automata and Cellular Learning Automata. Section 4 presents the proposed algorithm. Simulation results are given in section 5. Paper is concluded in section 6.

II. Particle Swarm Optimization

The particle swarm optimization simulates the behaviors of bird flocking. Suppose the following scenario; a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. The effective solution to find food is to follow the bird, which is nearest to the food. PSO learn from this scenario and use it to solve the optimization problems [1]. In PSO, each single solution is a bird in the search space that is called particle. All of the particles have fitness values, which are evaluated by a fitness function to be optimized. Particles have velocities, which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optimal solutions by updating generations. In each iteration, the velocity and the position of each particle i is updated using two quantities: the Personal Best solution obtained by particle i (PB_i) and the Global Best solution (GB) obtained by the group of particles. After finding these two quantities, particle i updates its velocity and its position according to the following equations.

$$v_{ij} = v_{ij} + rnd \times c_1 (PB_{ij} - x_{ij}) + rnd \times c_2 (GB_j - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

Where $v_i=(v_{i1}...v_{in})$ is the velocity of particle i, $x_i=(x_{i1},..., x_{in})$ is the current position (solution). rnd is a random number in the range (0,1). c_1 and c_2 are learning factors. Usually c_1 is equal to c_2. Particles velocities on each dimension are limited to a maximum velocity of V_{max}. If the sum of the accelerations causes the velocity on that dimension to exceed V_{max}, a parameter specified by the user, then the velocity on that dimension is set to V_{max}.

The first version of particle swarm algorithm reported by Kennedy in [1] operates in continuous search space. But many optimization problems are set in discrete space. For this reason in [2], Kennedy proposed a discrete binary version of the particle swarm (DPSO). DPSO is obtained by modifying equations in continuous PSO to be adapted to discrete binary space. In discrete binary space the search space can be viewed as a hypercube. Viewing the search space as a hypercube, the meanings of the concepts such as trajectory, velocity between and beyond used in continuous version of PSO will be changed. A particle may be seen to move nearer or farther from corners of hypercube by flipping various numbers of bits; in this way, velocity of the particles can be described by the number of bits changed per iteration. A particle with zeros bits changed does not move. A particle moves the farthest if all of its binary coordinates are changed. In this DPSO a dilemma occurs. What is the velocity or rate of change of a single bit or coordinate? Kennedy and Eberhart solved this dilemma by defining velocities and trajectories in term of changes in the probabilities that a bit will be 0 or 1.For binary discrete search spaces, DPSO updates the velocity according to equation (1) but computes the new position component to be 1 with a probability which is obtained by applying a sigmoid transformation $(1/(1+exp(-v))$ to the velocity component:

$$\begin{aligned} &\text{If } rnd < 1/(1+ e^{-v_{ij}}) \\ &\qquad x_{ij}=1 \\ &\text{else} \\ &\qquad x_{ij}=0 \end{aligned} \quad (3)$$

III. Cellular Automata, Learning Automata and Cellular Learing Automata

This section briefly describes Cellular Automata, Learning Automata and Cellular Learning Automata.

Learning Automata: Learning Automata are adaptive decision-making devices operating on unknown random environments. The Learning Automaton has a finite set of

actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA). In the following, the variable structure learning automata is described.

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

$$\begin{aligned} &\text{If } \beta(t)=0, \\ &\quad p_i(t+1) = p_i(t) + a[1 - p_i(t)] \\ &\quad p_{j\neq i}(t+1) = (1-a)p_j(t) \\ &\text{If } \beta(t)=1, \\ &\quad p_i(t+1) = (1-b)p_i(t) \\ &\quad p_{j\neq i}(t+1) = (b/s-1) + (1-b)p_j(t) \end{aligned} \tag{4}$$

Where a and b are reward and penalty parameters. When $a=b$, automaton is called L_{RP}. If $b=0$ and $0<b<<a<1$, the automaton is called L_{RI} and $L_{R\varepsilon P}$, respectively. For more information about learning automata the reader may refer to [9].

Cellular Automaton: Cellular Automata consists of a regular array of cells, each in one of finite number of states. The array can be in any finite number of dimensions. The state of a cell at time t+1 in CA is a function of the states of a finite number of cells (called its neighborhood) at time t. The simplest CA would be one-dimensional, with two possible states per cell, and a cell's neighbors defined to be the adjacent cells on either side of it. For more information about cellular learning automata the reader may refer to [13].

Cellular Learning Automata (CLA)**:** CLA which is obtained by combining cellular automata and learning automata is a mathematical model for dynamical complex systems that consists of a large number of simple learning components. Any number of LA can reside in a specific cell. Reinforcement signal for every automaton is computed according to CLA rule and actions of other LA residing in neighbor cells. This model has learning capability of LA and collective behavior and locality of CA. it is formally described in [6] as follows: d-dimensional CLA is a quintuple $CLA = (Z^d, \phi, A, N, F)$ that

- Z^d is a d-dimensional grid of cells. This grid can be finite, semi-finite or infinite.
- ϕ is a finite set of states that each cell can have.
- A is set of learning automata that each of them are assigned to a specific cell.
- $N = \{x_1,...,x_m\}$ is finite subset of Z^d that is called neighborhood vector.
- $F : \phi^m \rightarrow \underline{\beta}$ is local rule of CLA. $\underline{\beta}$ is set of valid reinforcement signals that can be applied to LA.

For more information about cellular learning automata the reader may refer to [6] [14] [15] [16] [17].

IV. A New Discrete PSO Based on Cellular Learning Automta

This section we first briefly describe LA-PSO and then present the proposed algorithm.

A. Discrete PSO based on learning automata

[3] introduced a discrete PSO that uses learning automata to compute probabilities of bits being zero or one. We call this model LA-PSO in the rest of the paper. In this model every particle has a set of learning automata that determine location of particle in the search space. Learning automata are reinforcement according to best location of each particle and the global best location. A learning automaton compares its current action (X_{ij}), corresponding bit in its personal best solution(PB_{ij}) and corresponding bit in global best solution(GB_j). If these bits are equal, automaton gets reward ($\beta=0$) otherwise gets punishment ($\beta=1$).

$$\begin{aligned} &\text{if } PB_{ij} = GB_{ij} = X_{ij} \\ &\qquad \beta_{ij}=0 \\ &\text{else} \\ &\qquad \beta_{ij}=1 \end{aligned} \tag{5}$$

This model uses L_{RP} learning schema. Suppose j'th learning automaton in particle i chooses action 0 or 1. A reinforcement signal will be generated as described above. Having reinforcement signal (β_{ij}) probabilities of j'th learning automaton in particle i will be updated using L_{RP} learning scheme that was mentioned previously in section III.

B. The proposed Algorithm: A cellular learning automata based Discrete PSO

In this section a new discrete PSO based on cellular learning automata is proposed. The proposed model is the combination of LA-PSO described before with cellular learning automata. In the proposed method every particle is assigned to one cell of a CLA with linear topology. Each cell of CLA is equipped with a set of learning automata. The set of learning automata in each cell (particle) determines the location (bits) of that particle in the search space. Figure 1 shows the placement of learning automata in cells (particles) of a CLA with linear topology. Each

particle in this configuration has some neighboring particle residing in the neighboring cells. If the neighborhood radius is r then a particle has $2r+1$particles as its neighbors, r particles to the left, r particle to the right and itself.

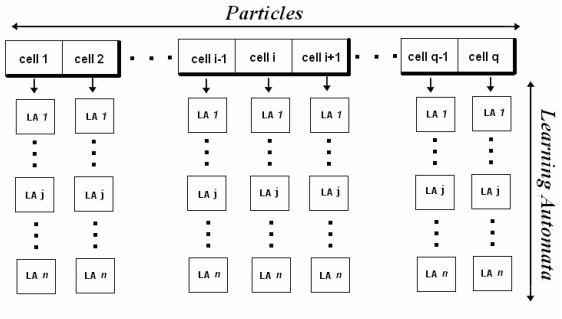

Fig. 1. Placement of LA in one dimensional CLA

Having CLA instead of LA in LA-PSO, the concept of neighborhood can be taken into consideration. In CLA-PSO each particle in a cell finds its local best (*LB*) by comparing the local bests of its neighbors and the global best (*GB*) paprmeter no longer exists. Each learning automaton j residing in cell i compares its action (X_{ij}) with the corresponding bits in its personal best (PB_{ij}) and local bests (LB_{ij}) *of* its neighbors. If these three bits are equal then reward signal is generated for that automaton ($\beta_{ij}=0$) otherwise punishment signal is generated ($\beta_{ij}=1$).

Two types of fixed and variable neighborhood are considered. In fixed neighborhood, radius of neighborhood (r) is set to be a fraction of the number of cells in CLA whereas in variable neighborhood, the radius of neighborhood increases from one to half of the array size in such a way that neighbors of a cell in the last generation is the set of all cells. That is, in the last generation the proposed algorithm operates exactly the same as LA-PSO. We call the proposed algorithm CLA-PSO when the neighborhood is fixed and VCLA-PSO when the algorithm it uses variable neighborhood.

We also introduce a new parameter called P_{Max}(>0.5). This parameter forces the probability of the value of a bit to be 1 or 0 in each learning automaton lie in the range [1-P_{Max},P_{Max}]. This pramater prevents premature convergence of LA-PSO that may happen because of early convergence of learning automata. It can also provide CLA-PSO with higher explorative ability in the late generations when learning automata are about to converge. The role of this parameter is similar to the role of mutation parobability in genetic algorithm. Note that in contrast to mutation probability in genetic algorithm, lower values of this parameter results in higher exploration strength.

In order to use CLA-PSO for an optimization (maximization for example) problem, first a set of learning automata is associated to each particle of CLA-PSO. The number of learning automata associated to a cell (and the corresponding particle) is the number of bits in the binary representation of points in the search space. Each automaton has two actions 0 and 1. The learning automaton, corresponding to the jth bit of the solution in cell i selects 1 with probability P_{ij} and 0 with probability 1-P_{ij}. Following steps will be repeated until a termination criterion is met.

1- Every automata in a cell i chooses one of its actions using its action probability vector
2- Particle i generates a new position X_i (a corner of the hypercube) by concatenating the actions chosen by its set of learning automata. If the fitness of new position is better than that stored in PB_i , PB_i will be updated.
3- Each particle computes its local best (LB_i) according to personal bests (PB) of its neighbors

$$LB_i = \{PB_j \mid j = \arg\max_{k \in N_i}(fitness(PB_k))\} \qquad (6)$$

Where N_i is set of topologic neighbors of cell i in CLA array. (*argmax* is used because it is assumed that we are solving a maximization problem).

4- According to PB_i and LB_i and X_i reinforcement signal $\beta_i=(\beta_{i1},...,\beta_{in})$ for learning automata of cell i is generated. If PB_{ij} and LB_{ij} and X_{ij} are equal then the jth learning automaton in cell i gets reward ($\beta_{ij}=0$) otherwise it gets punishment($\beta_{ij}=1$).
5- Reinforcement signa*l* β_i is used to update action probabilities of learning automata in particle i.
6- If P_{ij} is not in the range of [1-P_{Max} , P_{Max}] then.

$$P_{ij} = \min(\max(P_{ij}, 1-P_{\max}), P_{\max}) \qquad (7)$$

V. Simulation Results

In order to show the performance of the proposed algorithms five optimization problems are solved with CLA-PSO and VCLA-PSO and the results are compared with the results obtained with the results for discrete PSO (DPSO)[2] and LA-PSO[3]. Problems are two De Jong functions, Ackley function, Checker Board and Simple knapsack 0/1 problems. The test problems are briefly explained below:

De Jong Functions*:* De Jong functions used are borrowed from [7]. We call these two functions De Jong F1 and F2.

$$F_1(X) = \sum_{i=1}^{n} x_i^2 \quad |x_i| < 5.12$$

$$F_2(X) = \sum_{i=1}^{n-1} 100\,(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \quad |x_i| < 2.048$$

Ackley Function:

$$f_{ackley}(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right)$$

$$-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \quad |x_i| < 10$$

Checkerboard Problem: This problem is burrowed from [10]. In this problem a $s \times s$ grid is given. Each point of the grid can take two values 1 and 0. The goal of this problem is to create a checkerboard pattern of 0's and 1's on the

grid in such a way that each location with a value of 1 should be surrounded in all four directions by a value 0, and vise versa. The evaluation is measured by counting the number of correct surrounding bits for the present value of each bit position for a *(s-2)×(s-2)* grid. The chromosomes of this problem have dimension $n=s^2$. If we consider the grid as a matrix C=[c_{ij}] (*i,j=1,…,s*) and interpret $\delta(a,b)$ as the kronecher's delta function, the checkerboard function can be written as follows,

$$F_{CheckerBoard}(C) = 4(s-2)^2 - \sum_{i=2}^{n-1}\sum_{j=2}^{n-1}(\delta(c_{ij},c_{i-1\,j}) + \delta(c_{ij},c_{i+1\,j}) + \delta(c_{ij},c_{i\,j-1}) + \delta(c_{ij},c_{i\,j+1}))$$

Simple Knapsack 0/1 Problem*:* In the simple knapsack problem, there is a single bin of limited capacity *C*, and *n* elements of varying size. The problem is to select the elements that will yield the greatest usage bin capacity without exceeding it. If a solution selects too many elements, such that sum of elements' sizes become larger than bin capacity, some items are randomly removed from the bin until sum of elements' sizes become smaller than *C*.

$$f_{Knapsack}(X) = C - \sum_{i=1}^{n} x_i$$

Number of elements are 100 and the sizes of them are selected uniformly between intervals of [0,100]. C is set to 0.95 of sum of elements' sizes. Answers for this problem are binary strings of size 100. Each 1 in string indicates presence of corresponding element in bin. Details of test problems are given in table 1.

CLA-PSO radius of neighborhood is set to 1/20 of CLA array size. In VCLA-PSO radius of neighborhood grows linearly form 1 to half of CLA array size as generation number increases. P_{Max} in both CLA-PSO and VCLA-PSO is set to 0.95. Learning automata in all experiments use L_{RP} (a=b=0.1) learning scheme. DPSO is run with parameters reported in [2]. For each test problem population size is varied from 10 to 100 with step size of 5. The results are the average over 20 runs. Mean of best attained objective value in the last generation of algorithms over 20 runs for different population sizes are reported in figures 2 through 6

TABLE 1. DETAILS OF TEST PROBLEMS

Problem	*Knapsack*	*Ackley*	$F_{chekcher\ board}$	*De Jong* F_2	*De Jong* F_1
Decimal Dimension	-	10	-	10	10
Binary Dimension	100	50	100	50	50
Type	Min	Min	Max	Min	Min
Optimal Value	0	0	256	0	0
Number of Generation	100	500	500	500	500

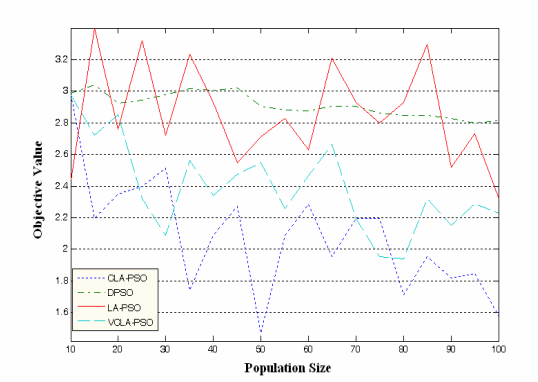

Fig. 2. Ackley function

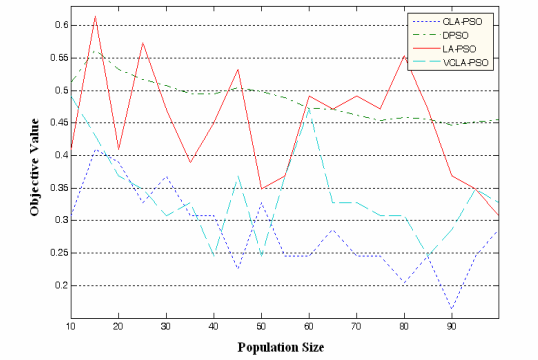

Fig 3. De Jong F1

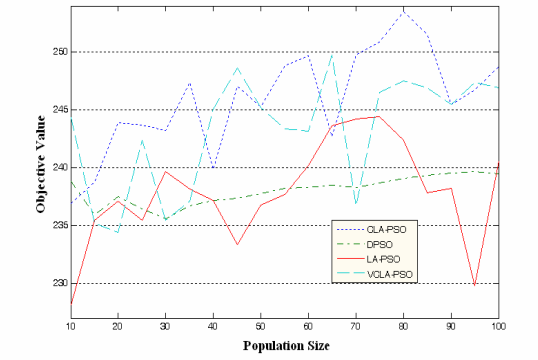

Fig. 4. Checker Board problem

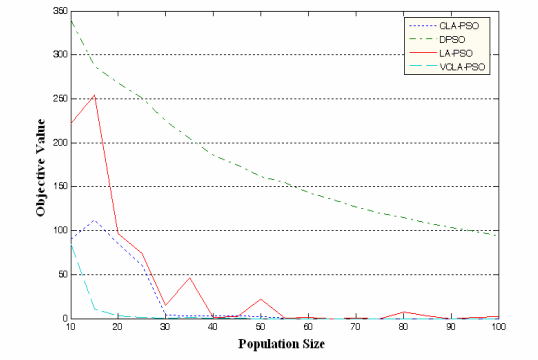

Fig. 5. Knapsack problem

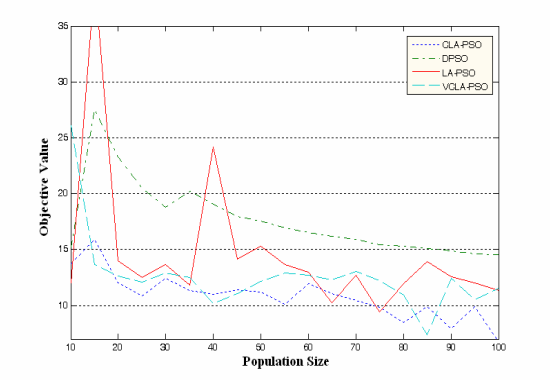

Fig. 6. De Jong F2

Figures 2 through 6 show that CLA-PSO outperforms other methods on Knapsack, Checker Board ، De Jong F1 and Ackley problems. VCLA-PSO performance is between CLA-PSO and LA-PSO in these problems while it produces better solutions on Knapsack problem compared to CLA-PSO. Slight improvement by (CLA-,VCLA-)PSO can be seen in De Jong F2. DPSO shows poor performance in function optimization compared to other discussed methods specially CLA-PSO

VI. Conclusion

In this paper in a new CLA based Discrete PSO was proposed. The proposed algorithm is a combination of the only reported learning automata based DPSO (LA-PSO) algorithm and cellular learning automata. Experimental results on five optimization problems show the superiority of the proposed algorithm.

References

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

[2] Kennedy, J., and Eberhart, R. C., "A Discrete Binary Version of The Particle Swarm Algorithm", Proceedings of Conference on Systems, Man, and Cybernetics, pp. 4104-4108, IEEE Service Center, Piscataway, NJ, 1997.

[3] Rastegar, R., Meybodi, M. R. and Badie, K., "A New Discrete Binary Particle Swarm Optimization based on Learning Automata", Proceedings of International Conference on Machine Learning and Applications (ICMLA2004), pp.456-462, USA, IEEE Press, 2004.

[4] Kennedy, J. and Mendes, R., "Population Structure and Particle Swarm Performance", Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002

[5] Suganthan, P. N., "Particle Swarm Optimizer with Neighborhood Operator", Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Piscataway, NJ, pp. 1958-1962, 1999

[6] Meybodi, M. R., Beigy, H., and Taherkhani, M., "Cellular Learning Automata", Proceedings of 6th Annual International Computer Society of Iran Computer Conference CSICC2001, Isfahan, Iran, pp. 153-163, 2001.

[7] De Jong, K. A., "The Analysis of the behavior of a class of genetic adaptive systems" Ph.D. dissertation, University of Michigan, Ann Arbor, 1975

[8] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[9] Narendra, K. S., and Thathachar, M. A. L., *Learning Automata: An Introduction*, Prentice-Hall Inc, 1989

[10] Baluja, S., Caruana, R., "Removing The Genetics from The Standard Genetic Algorithm", Proceedings of ICML'95, PP. 38–46, Morgan Kaufmann Publishers, Palo Alto, CA, 1995.

[11] Sheybani, M., Meybodi, M. R., "PSO-LA: A New Model for Optimization", Proceedings of 12th Annual International Computer Society of Iran Computer Conference CSICC2007, Iran, pp. 1162-1169, 2007.

[12] Sheybani, M. and Meybodi, M. R., "CLA-PSO: A New Model for Optimization", Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Volume on Computer, Telecommunication Research Center, Tehran, Iran, May 15-17, 2007.

[13] Wolfram, S., *Cellular Automata and Complexity*, Perseus Books Group, 1994.

[14] Beigy, H. and Meybodi, M. R., "A Mathematical Framework for Cellular Learning Automata", Advances on Complex Systems, Vol. 7, Nos. 3-4, pp. 295-320, September/December 2004.

[15] Rastegar, R., Meybodi, M. R. and Hariri, A., "A New Fine Grained Evolutionary Algorithm based on Cellular Learning Automata", *International Journal of Hybrid Intelligent Systems*, IOS Press, Volume 3, Number 2, pp. 83-98, 2006.

[16] Hariri, A., Rastegar, R., Zamani, M. S. and Meybodi, M. R., "Parallel Hardware Implementation of Cellular Learning Automata based Evolutionary Computing on FPGA", Proceedings of 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '05), pp. 311-314, California, USA, April 17-20, 2005.

[17] Masoodi, B., Meybodi, M. R. and Hashemi M., "Cooperative CLA-EC", Proceedings of 12th Annual CSI Computer Conference of Iran, Shahid Beheshti University, Tehran, Iran, pp. 558-559, Feb. 20-22, 2007.

[18] Shi, Y. and Eberhart, R. C., "A Modified Particle Swarm Optimizer", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[19] Wang, X. H. and Li, J. J., "Hybrid Particle Swarm Optimization with Simulated Annealing", Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, 2004.

[20] Gary, G. Y., and Haiming, Lu., "Dynamical Population Strategy assisted Particle Swarm Optimization", Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, Texas, pp. 697-702, October 2003.