Proceedings of the 2004 IEEE
Conference on Cybernetics and Intelligent Systems
Singapore, 1-3 December, 2004

A New Evolutionary Computing Model based on Cellular Learning Automata

R. Rastegar
Soft Computing Lab.
Computer Engineering Department
Amirkabir University
Tehran, Iran
rrastegar@ce.aut.ac.ir

M. R. Meybodi
Soft Computing Lab.
Computer Engineering Department
Amirkabir University
Tehran, Iran
meybodi@ce.aut.ac.ir

Abstract: In this paper, a new evolutionary computing model, called CLA-EC, is proposed. This new model is a combination of a model called cellular learning automata (CLA) and the evolutionary model. In this new model, each genome is assigned to a cell of cellular learning automata to each of which a set of learning automata is assigned. The set of actions selected by the set of automata associated to a cell determines the genome's string for that cell. Based on a local rule, a reinforcement signal vector is generated and given to the set learning automata residing in the cell. Based on the received signal, each learning automaton updates its internal structure according to a learning algorithm. The process of action selection and updating the internal structure is repeated until a predetermined criterion is met. This model can be used to solve optimization problems. To show the effectiveness of the proposed model it has been used to solve several optimization problems such as real valued function optimization and clustering problems. Computer simulations have shown the effectiveness of this model.

1. INTRODUCTION

Evolutionary algorithms form a class of random search algorithms in which principles of natural evolution are regarded as rules for optimization. They are often applied to optimization problem where specialized techniques such as gradient based algorithms, linear programming, dynamic programming, and etc, are not available or standard methods fail to give reasonable answers due to multimodality, nondifferentiability or discontinue of the problem at hand. The poor behaviors of evolutionary algorithms such as genetic algorithms in some problems, in which the designed operators of crossover and mutation do not guarantee that the building block hypothesis is preserved, have led to propose other approaches. Toward the development of a more robust evolutionary algorithm three main approaches have been taken to prevent building blocks disruption, in the first approach; researchers have focused on evolving a problem's genome representation in conjunction with its solution [1]. Others have attempted to evolve recombination operators using self-adaptation mechanisms [2]. And others have tried to replace the concept of recombination by explicitly modeling of good solutions in search space [3][4][5].

In [6], Cellular learning automata (CLA) which a combination of the cellular automata (CA) and learning automata (LAs) is introduced.. This model is superior to CA because of its ability to learn and also is superior to single LA because it is a collection of LAs, which can interact with each other toward solving a particular problem. The basic idea of CLA, which is super class of stochastic CA, is to use learning automata to adjust the state transition probability of stochastic CA [7]. So far, CLA have been used in many applications such as VLSI placement [8], rumor diffusion [9], channel assignment in cellular mobile systems [10], call admission control in cellular mobile system [11], cooperation in mutiagent systems [12]. For more information about CLA and its mathematical studies the reader may refer to [7]. In this paper a new model, called cellular learning automata based evolutionary computing (CLA-EC), which is the combination of cellular learning automata (CLA) and evolutionary computing model is presented. In this model, each genome is assigned to a cell of cellular learning automata to each of which a set of learning automata is assigned. The set of actions selected by the set of automata associated to a cell determines the genome's string for that cell. Based on a local rule, a reinforcement signal vector is generated and given to the set learning automata residing in the cell. Based on the received signal, each learning automaton updates its internal structure according to a learning algorithm. The process of action selection and updating the internal structure is repeated until a predetermined criterion is met. This model can be used to solve optimization problems. CLA-EC is capable of solving problems with very complex landscape. One of the advantages of CLA-EC like its counterpart, CLA, is its inherent parallelism.

The rest of this paper is organized as follows. Section 2 describes the learning automata and cellular learning automata. Section 3 introduces the CLA-EC algorithm. The experimental results of CLA-EC algorithm are proposed in Section 4. Finally we draw conclusion in Section 5.

2. CELLULAR LEARNING AUTOMATA

Learning Automata [13][14] 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. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action.

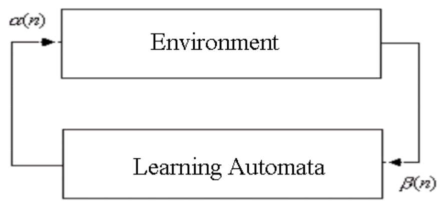


Fig. 1 The interaction between learning automata and environment

Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [19][28]. In the following, the variable structure learning automata is described. A VSLA is a quintuple $<\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{p},T(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{p})>$, where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{p}$ are an action set with s actions, an environment response set and the probability set $\boldsymbol{p}$ 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 $\boldsymbol{p}$ with respect to the performed action and received response. Let a VSLA operate in an environment with $\boldsymbol{\beta}=\{0,1\}$. Let $n \in N$ be the set of nonnegative integers. A general linear schema for updating action probabilities can be represented as follows. Let action i be performed at instance n.

If $\beta(n)=0$,

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1-p_i(n)] \\ p_{j\neq i}(n+1) &= (1-a)p_j(n) \end{aligned} \tag{1}$$

If $\beta(n)=1$,

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_{j\neq i}(n+1) &= (b/s-1) + (1-b)p_j(n) \end{aligned} \tag{2}$$

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. Figure 2 show the working mechanism of learning automata.

One of the models that are used to develop cellular evolutionary algorithm is a cellular automaton (CA). A cellular automaton is an abstract model that consists of large numbers of simple identical components with a local interaction. CA is non-linear dynamical systems which space and time are discrete in. It called cellular, because it is made up cells like points in the lattice or like squares of the checker boards and it is called automata, because it follows a simple rule [15]. The simple components act together to produce complicate patterns of behavior. CA performs complex computation with high degree of efficiency and robustness. It is especially suitable for modeling natural systems that can be described as massive collections of simple object interacting locally with each other. Cellular automaton has not only a simple structure for modeling complex systems, but also it can be implemented easily on SIMD processors. Therefore it has been used in evolutionary computing frequently. Mush literatures are available on cellular automata and its application to evolutionary computing, and the interested reader is referred to [16][15].

```
Initialize p to [1/s,1/s,…,1/s] where s is the number of actions
While not done
    Select an action i based on the probability vector p
    Evaluate action and return a reinforcement signal $\beta$
    Update probability vector using learning rule.
End While
```

Fig. 2 Pseudocode of variable-structure learning automaton

Cellular Learning Automata is a mathematical model for dynamical complex systems that consists of large number of simple components. The simple components, which have learning capabilities, act together to produce complicated behavioral patterns. A CLA is a CA in which learning automaton (multiple learning automaton) is assigned to its every cell. The learning automaton residing in particular cell determines its state (action) on the basis of its action probability vector. Like CA, there is a rule that CLA operate under it. The rule of CLA and the actions selected by neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in that cell. In CLA, the neighboring LAs of any particular LA constitute its local environment, which is nonstationary because it varies as action probability vector of neighboring LAs vary.

The operation of cellular learning automata could be described as follows: At the first step, the internal state of every cell specified. The state of every cell is determined on the basis of action probability vectors of learning automata residing in that cell. The initial value may be chosen on the basis of experience or at random. In the second step, the rule of cellular automata determines the reinforcement signal to each learning automaton residing in that cell. Finally, each learning automaton updates its action probability vector on the basis of supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained [7].

3. Cellular Learning Automata Based Evolutionary Computing

In this section, Cellular Learning Algorithm Based Evolutionary Computing, called CLA-EC is introduced as a new parallel model for evolutionary computing. In CLA-EC, similar to other evolutionary algorithms, the parameters of the search space are encoded in the form of genomes. Each genome has two components, model genome and string genome. Model genome is a set of learning automata. The set of actions selected by the set of learning automata determines the second component of the genome called string genome. For each cell, based on a local rule, a reinforcement signal vector is generated and given to the set of learning automata residing in that cell. Each learning automaton based on the received signal update its internal structure according to a learning algorithm. Then, each cell in CLA-EC generates a new string genome and compares its fitness with the fitness of the string genome of the cell. If the fitness of the generated genome is better than the quality of the sting genome of the cell, the generated string genome becomes the string genome of that cell. This process of generating string genome by the cells of the CLA-EC is iterated until a termination condition is satisfied. The main issue involved in designing a CLA-EC for a problem is finding a suitable genome representation and fitness function, and the parameters of CLA such as the number of cells (population size), topology and the type of the learning automata for each cell.

Evolutionary algorithms as the one described algorithm in this paper can be used in any arbitrary finite discrete search space. To simplify the algorithm, we assume that sight search space is a binary finite search space. So the optimization problem can be presented as follows. Assume $f:\{0,1\}^m \rightarrow \Re$ be a real function that is to be minimized. In order to use CLA-EC for the optimization function f first a set of learning automata is associated to each cell of CLA-EC. The number of learning automata associated to a cell of CLA-EC is the number bits in the string genome representing points of the search space of f. Each automaton has two actions called action 0 and 1. Then the 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- Cell *i* generates a new string genome, new^i, by combining the actions chosen by the learning automata of cell *i*. The newly generated string genome is obtained by concatenating the actions of the automata (0 or 1) associated to that cell. This section of algorithm is equivalent to learning from previous self-experiences.

3- Every cell *i* computes the fitness value of string genome new^i; if the fitness of this string genome is better than the one in the cell then the new string genome new^i becomes the string genome of that cell. That is

$$\xi^i_{n+1} = \begin{cases} \xi^i_n & f(\xi^i_n) \le f(new^i_{n+1}) \\ new^i_{n+1} & f(\xi^i_n) > f(new^i_{n+1}) \end{cases} \qquad (3)$$

4- *Se* cells of the neighboring cells of the cell *i* are selected. This Selection is based on the fitness value of the neighboring cells according to truncation strategy. This process is equivalent to mating in the nature. Note that mating in the context of proposed algorithm is not reciprocal, i.e., a cell selects another cell for mating but necessarily vise versa.

5- Based on selected neighboring cells a reinforcement vector is generated. This vector becomes the input for the set of learning automata associated to the cell. This section of algorithm is equivalent to learning from experiences of others. Let $N_s(i)$ be set selected neighbors of cell *i*. Define,

$$N_{i,j}(k) = \sum_{l \in N_s(i)} \delta(\xi^{l,j}_n = k), \qquad (4)$$

Where,

$$\delta(\exp) = \begin{cases} 1 & \exp \textit{ is true} \\ 0 & \textit{otherwise} \end{cases} \qquad (5)$$

$\beta^{i,j}$, the reinforcement signal given to learning automaton *j* of cell *i,* is computed as follows,

$$\beta^{i,j}_n = \begin{cases} u(N_{i,j}(1) - N_{i,j}(0)) & \textit{if } \xi^{i,j}_n = 0 \\ u(N_{i,j}(0) - N_{i,j}(1)) & \textit{if } \xi^{i,j}_n = 1 \end{cases} \qquad (6)$$

Where $u(.)$ is a step function. The overall operation of CLA-EC is summarized in the algorithm of figure of 3.

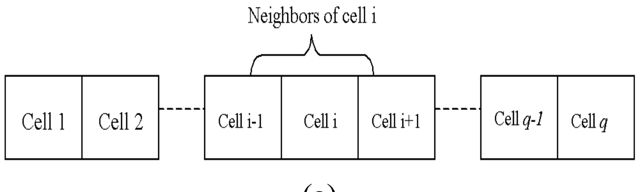
```
Initialize.
While not done do
    For each cell i in CLA do in parallel
        Generate a new string genome
        Evaluate the new string genome
        If f(new string genome)>f(old string genome) then
            Accept the new string genome
        End if
        Select Se cells from neighbors of cell i
        Generate the reinforcement signal vector
        Update LAs of cell i
    End parallel for
End while
```

Fig. 3 Pseudocode of CLA-EC

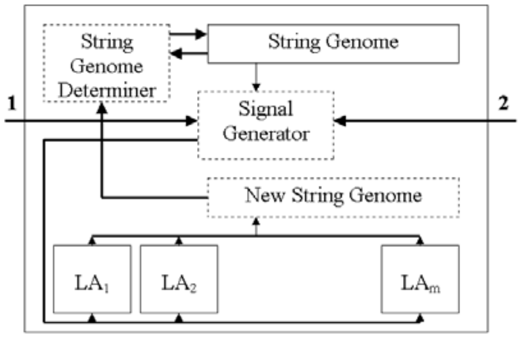
4. Simulation results

This section presents simulation results for five function optimization problems and the comparison of these results with the results obtained using Simple Genetic Algorithm (SGA), in terms of solution quality, and the number of function evaluations taken by the algorithm to converge completely for a given population size. The CLA-EC used for the experiments has a linear topology with wrap around connection as shown in figure 4a. The neighbors of cell *i* are cell *i-1* and cell *i+1*.

The architecture of each cell is shown in figure 4b. Each cell is equipped with *m* learning automata. The string genome determiner compares the new string genome with the string genome residing in the cell. The string with the higher quality replaces the string genome of the cell. Depending on the neighboring string genomes and the string genome of the cell, a reinforcement signal will be generated by the signal generator.



(a)



1:String Genome of Left Neighborhood
2: String Genome of Right Neighborhood

(b)

Fig. 4 a) A one-Dimensional (linear) cellular automaton with neighborhood radius one (*r=1*) and wrap around connection and *q* cells. b) The structure of a cell

Each quantity of the results reported is the average taken over 20 runs. The population size (number of cells in CLA-EC) varies from 3 to 49 with increments of two. The proposed algorithm is tested for learning algorithm L_{RP}. For the sake of convenience in presentation, we use *CLA-EC*(*automata(a,b),r,se,q)* to refer to the CLA-EC algorithm with *q* cells, neighborhood radius *r*, the number of selected cell *Se* when using learning automata *automata* with reward parameter *a* and penalty parameter *b.* The algorithm terminates when all learning automata converge completely. The proposed algorithms are tested five different standard function minimization problems. These functions that are given below are borrowed from reference [17].

$$F_1(X)=\sum_{i=1}^{3} x_i^2 \quad -5.12\le x_i \le 5.12$$

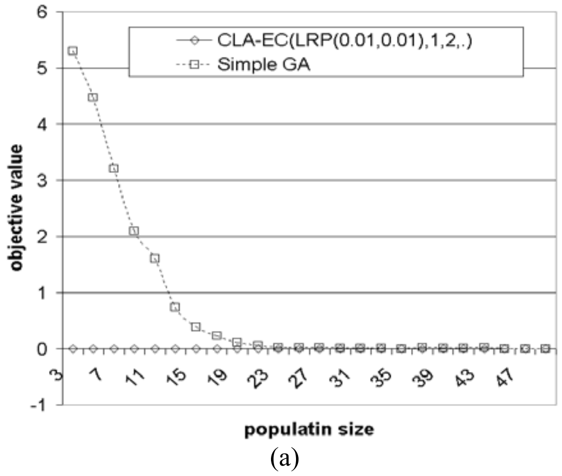
$$F_2(X)=100(x_1^2-x_2)^2+(1-x_1)^2 \quad -2.048\le x_i \le 2.048$$

$$F_3(X)=\sum_{i=1}^{5} \text{integer}(x_i) \quad -7.12\le x_i \le 7.12$$

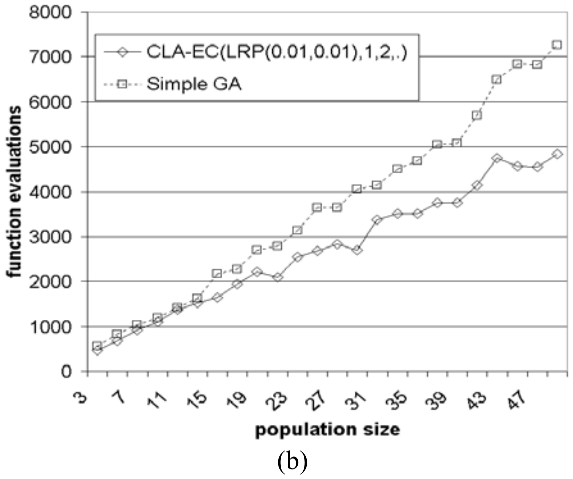
$$F_4(X)=\sum_{i=1}^{30} ix_i^4 + Gauss(0,1) \quad -1.28\le x_i \le 1.28$$

$$F_5(X)=(0.002+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1} \quad -65.536\le x_i \le 65.536$$

A simple genetic algorithm [18] that uses two-tournament selection without replacement and uniform crossover with exchange probability 0.5 is used in our experiments. Mutation is not used and crossover is applied with probability one. In this paper globally convergence is considered as termination condition for simple genetic algorithm. For functions F_1, F_2, F_4, F_5, we set *Se* to 2 because simulations have shown that value 2 for parameter *Se* is the most appropriate value for these functions. For F_3 function we set *Se* to 3. The results of comparisons are reported in figures 5 trough 9, which show the superiority of the proposed algorithm.

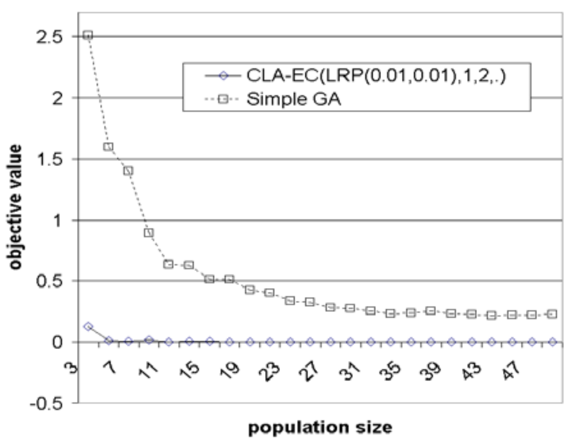


(a)

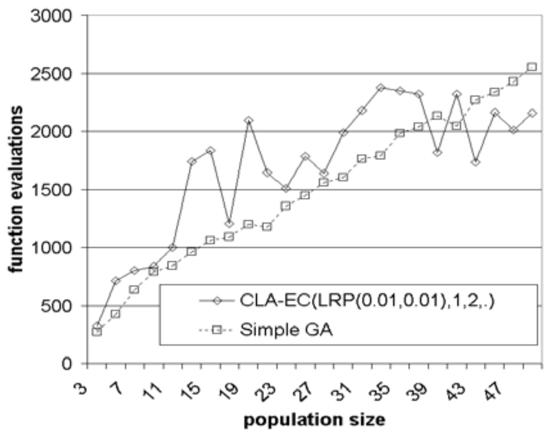


(b)

Fig. 5 CLA-EC(LRP(0.01,0.01),1, - ,5) and Simple GA for function F_1
a) Objective value b) function evaluations

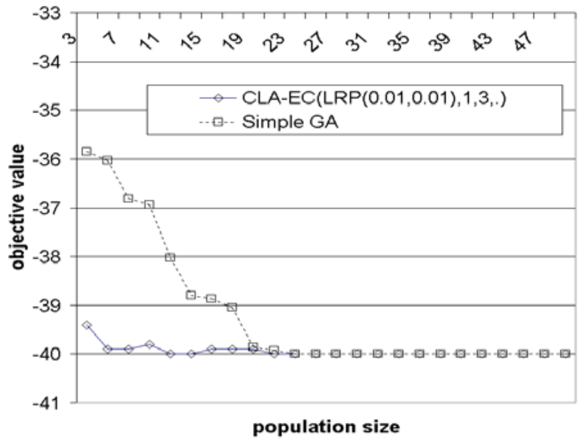


(a)

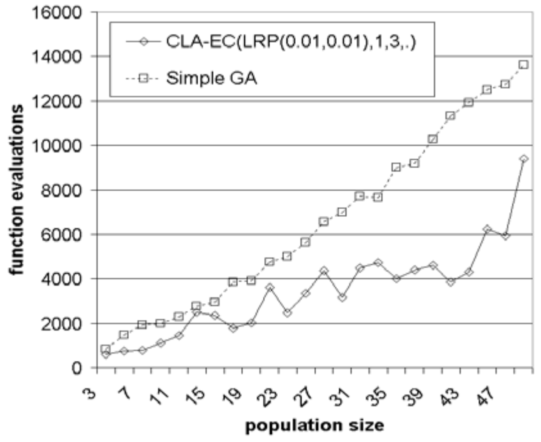


(b)

Fig. 6 CLA-EC(LRP(0.01,0.01),1, - ,5) and Simple GA for function F_2
a) Objective value b) function evaluations

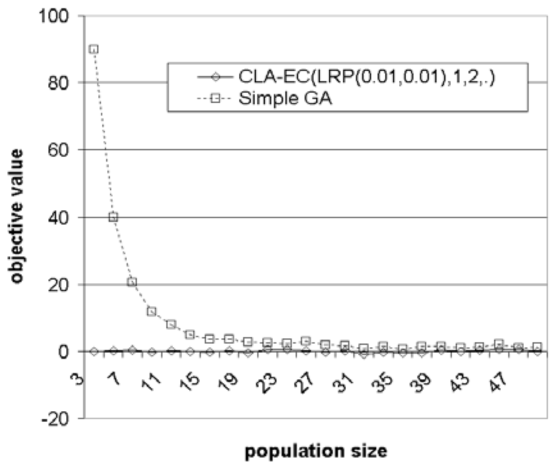


(a)

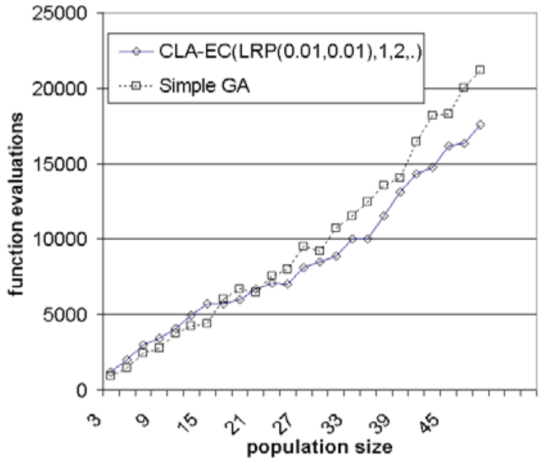


(b)

Fig. 7 CLA-EC(LRP(0.01,0.01),1, - ,5) and Simple GA for function F_3
a) Objective value b) function evaluations

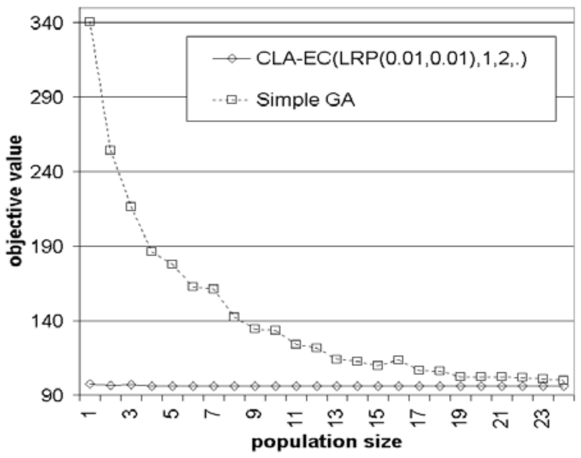


(a)

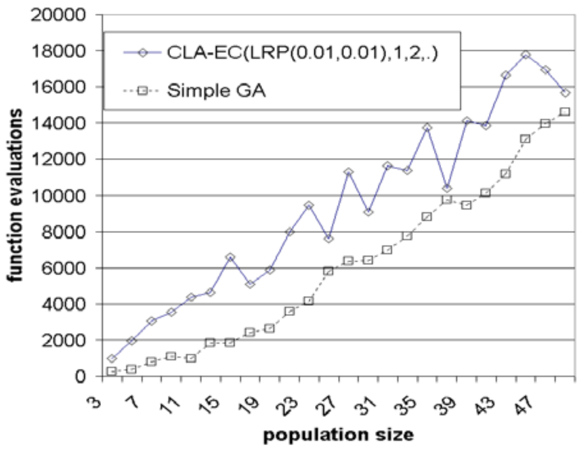


(b)

Fig. 8 CLA-EC(LRP(0.01,0.01),1, - ,5) and Simple GA for function F_4
a) Objective value b) function evaluation



(a)



(b)

Fig. 9 CLA-EC(LRP(0.01,0.01),1, - ,5) and Simple GA for function F_5
a) Objective value b) function evaluations

5. Conclusion

In this paper, the Cellular Learning Automata model is extended by combining with Evolutionary Computing Model and a new evolutionary model called CLA-EC proposed. The CLA-EC has a number of properties that make it superior over other evolutionary models. A highly degree of diversity is apparent in the early generations created by having the probabilities initially random and only slightly biased in the early iteration. In other hand with respect to the fact that interactions between cells (genomes) are local the probability of stuck in local optima can be decreased.

Reference

[1] Harik, G., "Learning Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms", Illinois Genetic Algorithm Report, No. 97005, Illinois University, Illinois, USA, 1997.

[2] Smith, J., and Fogarty, T. C., "Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm", In Proc. 3rd IEEE Conf. on Evolutionary Comp. IEEE Press, 1996.

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

[4] Baluja, S., and Davies, S., "Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of Search Space", Technical Report CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997.

[5] Mühlenbein, H., and Pelikan, M., "The Bivariate Marginal Distribution Algorithm", Advances in Soft Computing-Engineering Design and Manufacturing, PP. 521-535, 1999.

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

[7] Beigy, H., and Meybodi, M. R., "A Mathematical Framework for Cellular Learning Automata", Advanced in Complex Systems, to appear.

[8] Meybodi, M. R., and Mehdipour, F., "VLSI Placement Using Cellular Learning Automata", in Proceedings of 8th Annual International Computer Society of Iran Computer Conference CSICC2001, Mashhad, Iran, PP. 195-203, 2003.

[9] Meybodi, M. R., and Taherkhani, M., "Application of Cellular Learning Automata to Modeling of Rumor Diffusion", in Proceedings of 9th Conference on Electrical Engineering, Power and Water institute of Technology, Tehran, Iran, PP. 102-110, May 2001.

[10] Beigy, H., and Meybodi, M. R., "A Self-Organizing Channel Assignment Algorithm: A Cellular Leaning Automata Approach", Vol. 2690 of Springer-Verlag Lecture Notes in Computer Science, PP. 119-126, Springer-Verlag, 2003.

[11] Baradaranhashemi, A, Beigy, H., and Meybodi, M. R., "Dynamic Call Access Control for Cellular Mobile Networks", Proceedings of 9th Annual CSI Computer Conference, Computer Engineering Department, Sharif University, Tehran, Iran, pp. 440-446, Feb. 2004.

[12] Khojasteh, M. R. and Meybodi, M. R. "Cooperation in Multi-Agent Systems Using Learning Automata", Iranian Journal of Electrical and Computer Engineering, Vol. 1, No. 2, pp.81-91, 2004.

[13] Thathachar, M. A. L., Sastry, P. S., "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, PP. 711-722, 2002.

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

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

[16] Alba, E., and Troya, J. M., "Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms", Future Generation Computer Systems, Vol. 17, PP. 451-465, 2001.

[17] 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.

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