Recombinative CLA-EC

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

Abstract

Cellular Learning Automata (CLA) which is obtained by combining cellular automata (CA) and learning automata (LA) models is a mathematical model for dynamical complex systems that consists of a large number of simple learning components. CLA-EC, introduced recently is an evolutionary algorithm which is obtained by combining CLA and evolutionary computation (EC). In this paper CLA-EC with recombination operator is introduced. Recombination increases explorative behavior of CLA-EC and also provides a mechanism for partial structure exchange between chromosomes of population individuals that standard CLA-EC is not capable of performing it. This modification greatly improves CLA-EC ability to effectively search solution space and leave local optima. Experimental results on five optimization test functions show the superiority of this new version of CLA-EC over the standard CLA-EC.

1. Introduction

Cellular Learning Automata (CLA) first introduced in [1]. This model is obtained combining Cellular Automata [8] and Learning Automata [10][12] models. It is a mathematical framework for dynamic complex systems that consists of large number of learning components with local interactions. This model can solve problems with collective behavior of learning components, interacting with problem. This model have been applied to many problems such as channel assignment in cellular networks [2] , image processing [3], evolutionary computation [4], simulation of pesticide percolation [5], modeling rumor diffusion [6], VLSI placement [7]. As mentioned above an application of CLA is evolutionary computing. Cellular Learning Automata based Evolutionary Computing (CLA-EC) [4] is a newly proposed algorithm in this area. In this model every cell contains one chromosome. Each cell selects its chromosome according to its LA's actions. A reinforcement signal according to actions of the LA and the neighborhood topology is generated. This model is capable of performing search in complex, large and multimodal landscapes. This model suffers from lack of explorative behavior. This problem is solved with the use of recombination borrowed from genetic algorithm.

Genetic Algorithm [9] is a search algorithm based on the mechanics of natural selection and natural genetics. It has some operators that mimic natural genetic operators: Selection, Recombination and Mutation. Selection ensures survival of fittest solution in population. Recombination is a genetic operator used to generate children chromosomes from parent chromosomes. Crossover or recombination is inheriting genetic information of parents by children. A child has some of the mother and some features of father. Recombination is done with the hope that children inherit good features of parents and generating individuals that are better than parents are. It is an analogy to reproduction and biological recombination. Mutation is a genetic operator used to maintain genetic diversity and it is analogous to biological mutation.

The poor behaviors of evolutionary methods 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 the development of Estimation of Distribution Algorithms (EDA) [15]. Toward the development of a more robust evolutionary algorithm some approaches have been taken to prevent building blocks disruption. [4][15], [16][17] to name a few. All of these methods do not use recombination because of its disruption in building blocks. In contrast some literatures discuss benefits of using recombination and that recombination can

provide better diversification and wider exploration of search space [18][19].

In this paper we introduce a new version of CLA-EC that uses recombination operator. This modification does improve its ability in leaving local optima and effectively exploring the search space without adding much overhead on computation needs. The new version of CLA-EC combines adjacent chromosomes (called string genomes in CLA-EC) and generates new chromosomes using shuffle recombination. In order to demonstrate the effectiveness of this version of CLA-EC, five optimization test functions are used for comparison. Simulation results show the superiority of the improved CLA-EC over standard CLA-EC.

The rest of paper is organized as follows: Cellular Learning Automata is described in section 2. Section 3 briefly presents CLA-EC. In section 4 recombinative CLA-EC is described. Simulation results are given in section 5. Paper is concluded in section 6.

2. Cellular Learning Automata

In this section Cellular Automata, Learning Automata and Cellular Learning Automata are briefly introduced.

A *cellular automaton* 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 is 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 automata the reader may refer to [8][20].

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 choosing 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 with the environment can be made to result in the selection of the optimal action. 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 (FSLA) and variable structure learning automata (VSLA). In the following the variable structure learning automata is described.

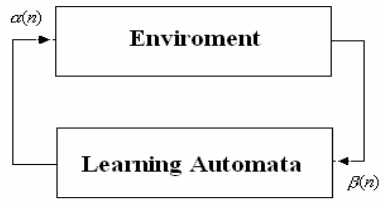


Figure 1. The interaction between learning automata and environment

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, environment response set and the probability set 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 (Reinforcement Signal). Let a VSLA operate in an environment with $\boldsymbol{\beta}=\{0,1\}$. Let $t \in N$ be the set of nonnegative integers representing step number. A general linear schema for updating action probabilities can be represented as follows. Let action i be performed.

If $\beta(t)=0$,

$$p_i(t+1) = p_i(t) + a[1 - p_i(t)]$$

$$p_{j \neq i}(t+1) = (1-a)\, p_j(t)$$

If $\beta(t)=1$,

$$p_i(t+1) = (1-b)\, p_i(t)$$

$$p_{j \neq i}(t+1) = (b/s - 1) + (1-b)\, p_j(t)$$

Where a and b are reward and penalty parameters. When $a=b$, automaton is called L_{RP}. If $b=0$ or $0<b<<a<1$, the automaton is called L_{RI} or $L_{R\varepsilon P}$, respectively. The overall operation of learning automaton is summarized in the algorithm of figure 2. For more information about learning automata the reader may refer to [10][12][13][14].

Initialize **p** to [1/s,…,1/s] (s is the number of actions and **p** is actions' probability vector)
While not done
  Select an action *i* based on the vector **p**
  Evaluate the action and return $\boldsymbol{\beta}$
  Update **p** using the learning algorithm
End While

Figure 2. Pseudocode of VSLA

Cellular Learning Automata (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 LA 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. 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. 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. For more information about cellular learning automata the reader may refer to [1][21].

3. Cellular Learning Automata based Evolutionary Computing (CLA-EC)

The CLA-EC model [4] is obtained by combining cellular learning automata and evolutionary computing. This model is capable of performing search in complex, large and multimodal landscapes. 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 (chromosome). Model genome is a set of learning automata. The set of actions selected by this set of learning automata determines the second component of genome (string genome). Based on a local rule, a reinforcement signal vector is generated and given to the set of learning automata. According to the learning algorithm, each learning automaton in the set of learning automata updates its internal state. Then each learning automata in a cell chooses one of its actions using its probability vector. The set of actions chosen by the set of automata residing in a cell determines a candidate string genome (chromosome) that may replace the current string genome. The fitness of this string genome is then compared to the fitness of the string genome residing in that cell. If the fitness of the generated genome is better than the quality of the string genome of the cell, the generated string genome becomes the string genome of that cell. The process of generating string genome by the cells of the CLA-EC is repeated until a termination condition is satisfied. In order to have an effective algorithm, the designer of the algorithm must be careful about determining a suitable genome representation, fitness function for the problem at hand, the parameters of CLA such as the number of cells (population size), the topology, and the type of the learning automata for each cell. Assume $f:\{0,1\}^m \to R$ be a real function that is to be minimized. In order to use CLA-EC for optimization of function *f*, first a set of learning automata will be assigned to each cell of CLA-EC. The number of learning automata assigned to a cell of CLA-EC is the number of bits in the string genome representing points of the search space of *f*. Each automaton has two actions: 0 and 1. The CLA-EC iterates the following steps until the termination condition is met.

Step 1- Every automata in a cell *i* chooses one of its actions using its action probability vector

Step 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.

Step 3- Every cell *i* computes the fitness value of string genome new^i; if the fitness of this string genome is better than the string genome in the cell (ξ^i) then the new string genome new^i will be replaced with ξ^i. That is

$$\xi_{n+1}^i = \begin{cases} \xi_n^i & f(\xi_n^i) \le f(new_{n+1}^i) \\ new_{n+1}^i & f(\xi_n^i) > f(new_{n+1}^i) \end{cases}$$

Step 4- *Se* neighbors 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.

Step 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 of selected neighbors of cell *i*. Define,

$$N_{i,j}(k) = \sum_{l \in N_s(i)} \delta(\xi_n^{l,j} = k)$$

Where

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

$\xi_n^{l,j}$ is *j*th bit of string genome residing in cell *l* at time *n*. $\beta_{i,j}$, the reinforcement signal given to learning automaton *j* of cell *i,* is computed as follows,

$$\beta_n^{i,j} = \begin{cases} u(N_{i,j}(1) - N_{i,j}(0)) & if \ \xi_n^{i,j} = 0 \\ u(N_{i,j}(0) - N_{i,j}(1)) & if \ \xi_n^{i,j} = 1 \end{cases}$$

Where $u(.)$ is a step function. Pseudo code of this algorithm is given in section 4. For more information about CLA-EC the reader may refer to [4][22][23][24][25].

4. Recombinative CLA-EC

During the operation of standard CLA-EC, learning automata residing in each cell change their probability vectors in such a way that produce locally good string genomes in its neighborhood for the next generations. All bits of these locally good string genomes are assumed to be good and will be used to learn learning automata. There might be some string genomes that have not high overall quality but contain good partial solutions. Standard CLA-EC has no mechanism for partial string exchange between cells (population) that contain theses strings. String genomes in CLA-EC that have partially good answers but do not have overall high quality are not selected in search process and their valuable information will not be transferred to the next generation. This model also suffers from lack of exportation behavior. To solve these problems we use recombination to manipulate string genomes residing in cells.

Recombination of string genomes in CLA-EC has two benefits: 1.Better exploration of search space: Recombination can cause dramatic changes in string genomes and therefore better exploration of search space. 2. Partial chromosomes exchanges between cells with the hope that generated offspring inherit good features of parents. For example, consider two adjacent cells that contain partially good solutions. They do not have overall high quality so they are not selected in step 4(Selection Step) of CLA-EC and their valuable information in the next generation will be destroyed. But recombination can selects good parts of these two string genomes probabilistically and put them together to produce overall high quality solutions.

In one point recombination crossover point on the parent chromosomes is randomly selected. All genes after that point in the chromosomes are swapped between the two parent chromosomes. The resulting chromosomes are the children. *Shuffle recombination* [11] which is used in this paper is similar to one point recombination: A crossover position is randomly selected. However, before the genes are exchanged, they are randomly shuffled in both parents. After recombination, the genes in the children are un-shuffled. This removes positional bias as the variables are randomly reassigned each time recombination is performed.

In the modified CLA-EC every cell *i* (i is even) in each time step recombines its string genome (chromosome) with that of cell *i+1*. Then the string genome of cells *i* and *i+1* are replaced by the genomes generated by the recombination operation. Of course, If the new string genomes chosen by the learning automaton of a cell are better (worse) than the genomes resulted from the recombination in the last step then the existing string genomes in the cell will be (not be) replaced. This way neighboring genomes exchange some parts of their structure and benefit form each other partial solutions. Recombinative CLA-EC algorithm is summarized in the algorithm of figure 3. We call this modified algorithm, **R**ecombinative CLA-EC (**R**CLA-EC). Pseudo code of CLA-EC is like RCLA-EC without lines 14-17 in figure 3.

```
1: Initialize.
2: While not done do
3:   For each cell i in CLA do in parallel
4:     Generate a new string genome
5:     fNew(i)=fitness(new genome of cell i)
6:     If fNew(i) < fOld(i) then
7:       Accept the new string genome
8:       fOld(i) = fNew(i)
9:     End if
10:    Select Se cells from neighbors of cell i
11:    Generate the reinforcement signal vector
12:    Update learning automata of cell i
13:  End parallel for
14:  For each cell i in CLA do in parallel
15:    Cell i (i is even) recombines its string
       genome with that of cell i+1 with a fixed
       probability and produces two strings that are
       exchanged with strings in cells i and i+1.
16:    fOld(i)=fitness(new string genome of cell i)
17:  End parallel for
18:End while
```

Figure 3. **Pseudocode of RCLA-EC**

5. Simulation Results

In order to show the performance of the proposed algorithm, it has been tested on five function minimization problems and then the results are compared with the results obtained using standard CLA-EC. These functions are Ackley, Schwefel, Rastrigin, Griewangk, and Rosenbrock functions which are given in table 1. In this table, considered range, optimum point (x^*) and formula for each

Table 1. Test functions

Function name	Considered range	x^* $f(x^*)$	Formula
Ackley	$-10 \le x_i \le 10$	(0,0,...0) 0	$f(X) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
Rastrigin	$-5.12 \le x_i \le 5.12$	(0,0,...0) 0	$f(X) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$
Schwefel	$-512.03 \le x_i \le 512.03$	(420.9687,...,420.9687) 0	$f(X) = 418.9829\, n + \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$
Griewangk	$-600 \le x_i \le 600$	(0,0,...0) 0	$f(X) = 1 + \sum_{i=1}^{n} \frac{x_i}{4000} - \prod_{i=1}^{n} \frac{x_i}{\sqrt{i}}$
Rosenbrock	$-2.048 \le x_i \le 2.048$	(1,1,...1) 0	$f(X) = \sum_{i=1}^{n-1} 100\,(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$

function are given. These Functions are all highly multimodal and have dimension of 20. Two experiments are conducted to show the effectiveness of the new algorithm.

Five bits are used for encoding each dimension. It means that every cell contains 100 LA. Maximum number of generations for all problems is set to 1500. Reward and penalty parameters (a and b) of LA as reported in [4] are set to 0.01. The CLA-EC and RCLA-EC used for the experiments have linear topology with wrap around connection. The neighbors of cell i are cell i-1, cell i+1 and itself. Learning algorithm for learning automata is L_{RP}. Se in both algorithms is set to 2. In RCLA-EC every even numbered cell (in zero based numbering) does recombination with next odd numbered cell with probability of 0.7.

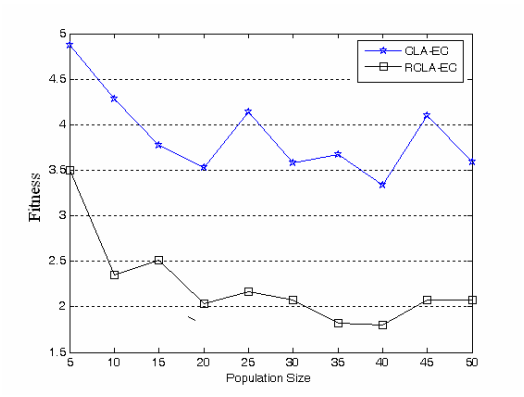
In the first experiment population size is varied form 5 to 50 with step size 5. Figures 4 to 8 show the mean and variance of the best fitness in the last generation of CLA-EC and RCLA-EC over 20 runs. The results show that the mean of the best obtained fitness using RCLA-EC is always better than that of CLA-EC. The variance of the best obtained fitness also is smaller in most cases. As the population size increases in RCLA-EC, the variance goes to zero. This is a desired behavior because it means that algorithm do not produce supervening result.

The second experiment is conducted to show the effect of recombination operator on the process of search, made by RCLA-EC. For this purpose we used RCLA-EC without learning step (step 12 in pseudocode of figure 3) and called this modified algorithm "RCLA-EC-WL". Algorithm RCLA-EC-WL will be used to only investigate the effect of recombination on the searching capability of CLA-EC. The results obtained for RCLA-EC-WL are compared with the results obtained for CLA-EC and RCLA-EC. Figure 9 shows the mean and the variance of population fitness during the process of search by CLA-EC, RCLA-EC and RCLA-EC-WL with population size of 10 for Schwefel function. Results are averaged over 50 runs. All other functions of table 1 and other population sizes generate the same results with nominal differences.

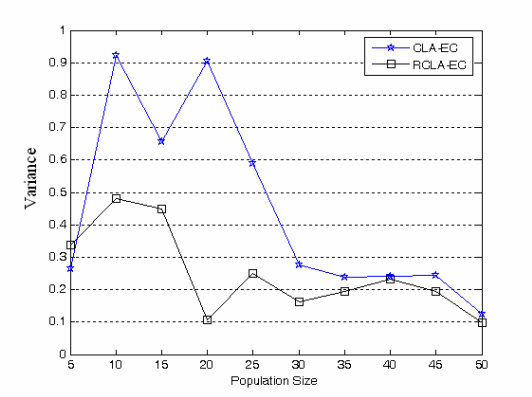
From figure 9.a one can conclude that the recombination operator increases the explorative strength of RCLA-EC during the early stages of the search and its exploitative strength during the late stages of the search. This causes RCLA-EC to converge to a better solution than CLA-EC is able to find. The mean fitness of RCLA-EC starts with a lower slope than CLA-EC which means that the recombination operator prevents the RCLA-EC from quickly tapping in local optima. Figure 9.a also shows that RCLA-EC-WL can not improve its population as the search proceeds. This is because the variance of fitness does not change much during the search process causing RCLA-EC-WL fails to find an acceptable solution.

Figure 9.b shows that the variance of fitness for CLA-EC drops rapidly in the early stages of search process and causes the algorithm to trap into a local optima. This figure also shows that on the contrary,

RCLA-EC continues its search process with a more diverse population leading to better exploration during the early stages of search process. Moreover, this figure clarifies that the exploitation capability of RCLA-EC is more than that of CLA-EC in the late stages of the search process resulting in more accurate results.

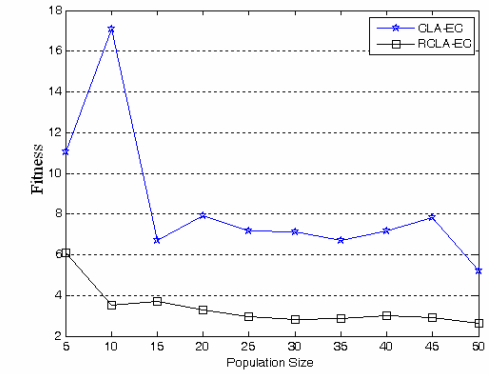


(a) Best fitness mean

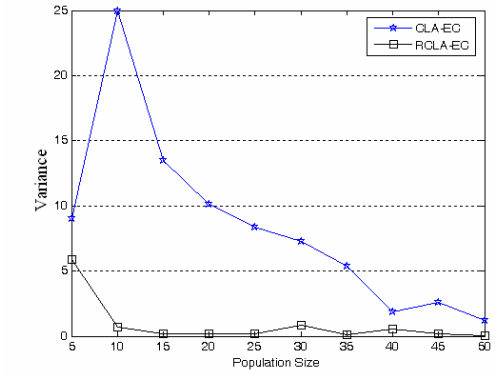


(b) Best fitness variance

Figure 4. CLA-EC and RCLA-EC for Ackley function

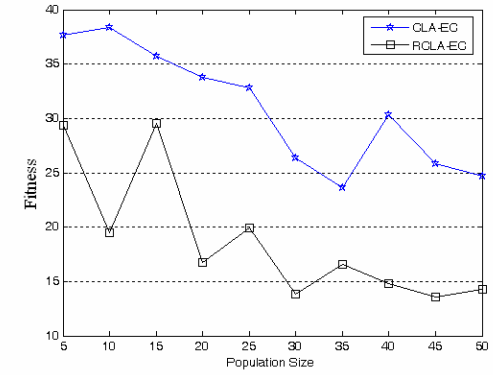


(a) Best fitness mean

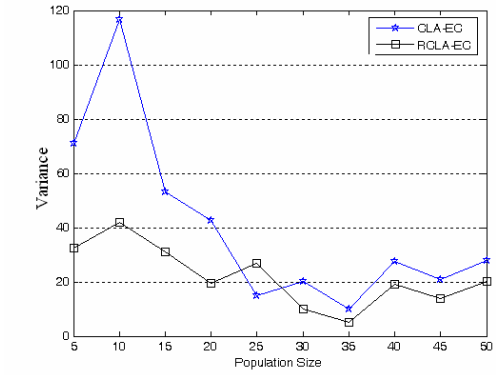


(b) Best fitness variance

Figure 5. CLA-EC and RCLA-EC for Griewangk function

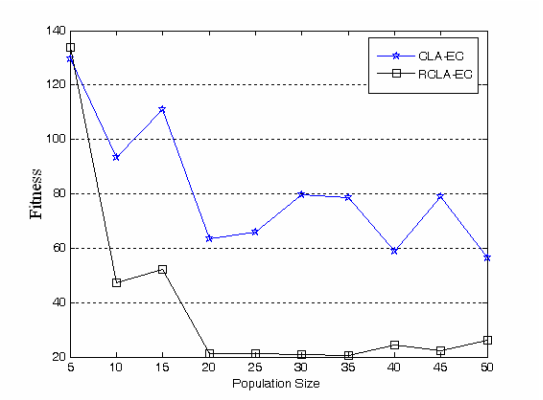


(a) Best fitness mean

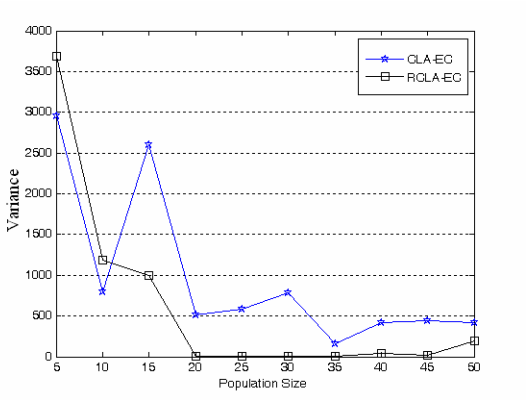


(b) Best fitness variance

Figure 6. CLA-EC and RCLA-EC for Rastrigin function

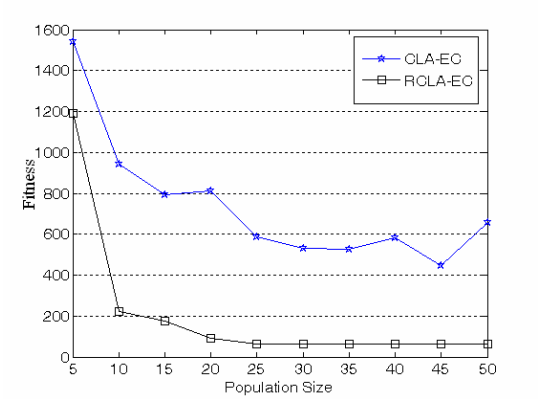


(a) Best fitness mean

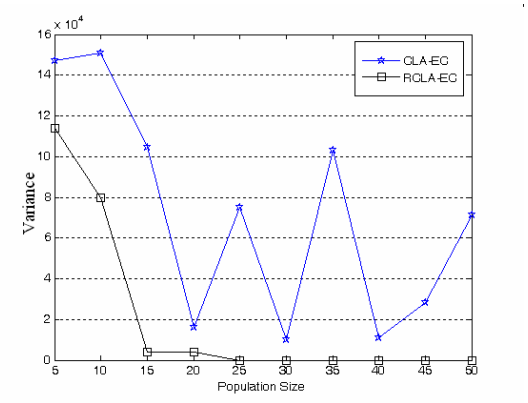


(b) Best fitness variance

Figure 7. CLA-EC and RCLA-EC for Rosenbrock function

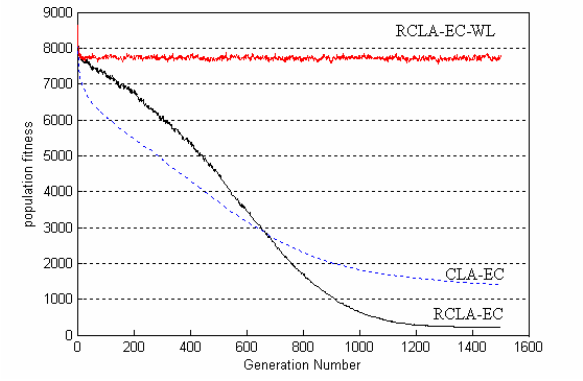


(a) Best fitness mean

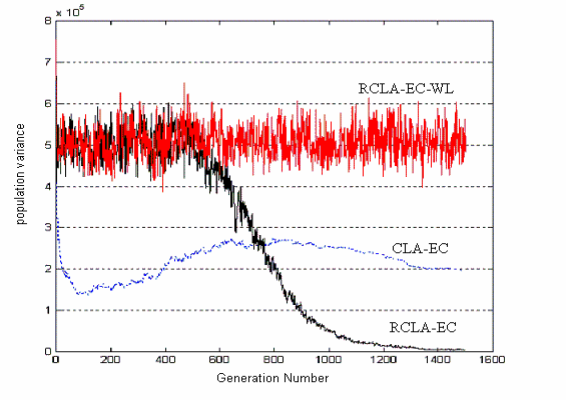


(b) Best fitness variance

Figure 8. CLA-EC and RCLA-EC for Schwefel function



(a) Mean of population fitness



(b) Variance of population fitness

Figure 9. CLA-EC and RCLA-EC for Schwefel function (Population Size = 10)

6. Conclusion

This paper introduced a modified version of CLA-EC that uses recombination. Recombination increases explorative behavior of CLA-EC and also provides a mechanism for partial structure exchange between chromosomes of individuals that standard CLA-EC is not capable of performing it. This modification greatly improved CLA-EC ability to effectively search solution space and escaping from local optima. In order to show the performance of the proposed algorithm, it is tested on five optimization test functions. Simulation results showed the superiority of the modified CLA-EC in optimization of multimodal functions compared to standard CLA-EC.

7. References

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

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

[3] M.R. Meybodi, and M.R. Kharazmi, "Image Restoration Using Cellular Learning Automata", Proceedings of the Second Iranian Conference on Machine Vision, Image Processing and Applications, Tehran, Iran, 2003, pp. 261-270.

[4] R. Rastegar, and M.R. Meybodi, "A New Evolutionary Computing Model based on Cellular Learning Automata", Proceedings of 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS), Singapore 1-3 Dec. 2004, pp. 433-438.

[5] M.R. Meybodi, and S. Noferesty, "Simulation of Pesticide Percolation in the soil Based on Cellular Learning Automata", Proceedings of 12th Annual International Computer Society of Iran Computer Conference CSICC2007, Tehran, Iran, 2007, pp. 2247-2251.

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

[7] M.R. Meybodi, and F. Mehdipour, "VLSI Placement Using Cellular Learning Automata", *Journal of Modares*, University of Tarbeit Modares, Vol. 16, summer 2004, pp. 81-95.

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

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

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

[11] F.J. Burkowski, "Shuffle Crossover and Mutual Information", Proceedings of the Congress on Evolutionary Computation, 1999, pp. 1574-1580.

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

[13] M.A.L. Thathachar, and. K.R. Ramakrishnan, "A hierarchical system of learning automata" *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, 1981 pp. 236-241

[14] B.J. Oommen, and M. Agache, "A Comparison of Continuous and Discredited Pursuit Learning Schemes", Technical Report, School of Computer Science, Carleton University, Canada, Ottawa, K1S 5B6, 1999.

[15] Larranaga, P., and J.A. Lozano, *Estimation of Distribution Algorithms. A New tools for Evolutionary Computation*, Kluwer Academic Publishers, 2001.

[16] P. Larranaga, R. Etxeberria, J.A. Lozano, and J.M. Pena, "Optimization by Learning and Simulation of Bayesian and Gaussian Networks", Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence, University of Basque Country, December 1999.

[17] M. Pelikan, D.E. Goldberg, and F. Lobo, "A Survey of Optimization by Building and Using Probabilistic Model", Illinois Genetic Algorithm Report, No. 99018, Illinois University, Illinois, USA, 1999.

[18] W.M. Spear, and K.A. De Jong, "An analysis of multipoint crossover," in Foundations of Genetic Algorithms, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 301–315.

[19] L.J. Eshelman, and J.D. Schaffer, "Crossover's niche," in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 9–14.

[20] Gutowitz, H., *Cellular Automata: Theory and Experiment,* MIT Press, Cambridge, 1991.

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

[22] A. Hariri, R. Rastegar, M.S. Zamani, and M.R. Meybodi, "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), California, USA, April 17-20, 2005, pp. 311-314.

[23] B. Masoodifar, M.R. Meybodi, and R. Rastegar, "Asynchronous CLA-EC", Proceedings of 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab., Tehran, Iran, Jan. 24-26, 2006, pp. 447-458.

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

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