On Expediency of Closed Asynchronous Dynamic Cellular Learning Automata

Ali Mohammad Saghiri, and Mohammad Reza Meybodi

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran

Abstract: Closed Asynchronous Dynamic Cellular Learning Automata (*CADCLAs*) have been reported recently. These models are hybrid models based on Learning Automata (*LAs*) and Cellular Automata (*CAs*). Because of distributed computation characteristic of *CAs* and probabilistic decision making nature of *LAs*, analyzing the performance of *CADCLA* based algorithms is difficult. The expediency metrics has been used to study the behavior of the *LA* based models. With respect to this metric, the *CADCLAs* have not been studied in the literature. In this paper, a set of conditions under which a *CADCLA* is called expedient will be suggested.

***Index Terms*— Closed Asynchronous Dynamic Cellular Learning Automata, Expediency.**

I. INTRODUCTION

Cellular Automata (*CAs*) are computational models which composed of independent and identical cells. In these models, the cells are arranged into a lattice. In a *CA,* each cell selects a state from a finite set of states. A cell uses the previous states of a set of cells, including the cell itself, and its neighbors and then updates its state using a rule called local rule. *CAs* evolves in discrete time steps [1], [2]. On the other hand, Learning Automata (*LAs*) are models for adaptive decision making in unknown environments. A set of actions has been defined for this model. Each action has a probability which is unknown for the *LA* for getting reward by the environment. This model tries to find an appropriate action through repeated interaction with the environment. The appropriate action is an action with the highest probability of getting reward by the environment.

Cellular Learning Automata (*CLAs*) are hybrid models based on *CAs* and *LAs*[3]. These models inherit the computational power from *CAs* and the learning capability in unknown environment from *LAs*. A *CLA* is a *CA* in which a *LA* is assigned to each cell. The *LA* residing in a particular cell determines its state (action) according to its action probability vector. Like *CA*, there is a local rule that the *CLA* operates under. The local rule of the *CLA* and the actions selected by the neighboring *LA*s of any particular *LA* determine the reinforcement signal to that *LA*. The neighboring *LA*s (cells) of any particular *LA* (cell) constitute the local environment of that *LA* (cell). *CLAs* have found applications in areas such as computer networks[4]–[7], social networks [8], Petri nets[9], and evolutionary computing [10] to mention a few. Recently, dynamic models of *CLAs (DCLAs)* have been reported in [11], [4], and [12].

CLAs can be classified into two main classes: *SCLAs* and *DCLAs* described as follows. In a *SCLA*, the structure of the cells remains fixed during the evolution of the *CLA* [8], [13]–[17]. In a *DCLA*, one of its aspects such as structure, local rule, attributes or neighborhood radius may change over time. *DCLAs* can also be classified as *Synchronous DCLAs (SDCLAs)* or *Asynchronous DCLAs (ADCLAs).* In *SDCLAs,* all *LAs* in different cells are activated synchronously whereas in *ADCLA*s the *LAs* in different cells are activated asynchronously. All the reported *DCLAs* are asynchronous [11],[4], [12]. *DCLAs* can be either open or closed. In closed *DCLAs*, the states of neighboring cells of each cell called local environment affect on the action selection process of the *LA* of that cell whereas in open *DCLAs*, each cell, in addition to its local environment has an exclusive environment which is observed by the cell only and the global environment which can be observed by all the cells.

The expediency metric has been widely used to study the behavior of the *LA* based models such as *CLAs* in the literature[18]. A learning automaton that performs better than its equivalent pure–chance automaton is said to be expedient. In a pure-chance automaton, the actions of the automaton are always selected with equal probabilities. A *CLA* model is expedient when it performs better than its equivalent pure-chance model in which the *LA* of each cell is replaced with a pure-chance automaton. Because of distributed computation characteristic and dynamicity of *DCLA*s, analyzing the performance of *DCLAs* is difficult. It should be noted that, the expediency of *Closed Asynchronous DCLAs (CADCLAs)* has not been studied in the literature.

In this paper, a set of conditions under which a *CADCLA* is called expedient will be suggested. The rest of the paper is organized as follows. In section 2, the learning automata used in this paper is described. Section 3 reviews the *CADCLAs*. Section 4 is dedicated to required definitions. In section 5, the conditions under which a *CADCLA* is called expedient are proposed. In order to support the proposed results, a computer simulation is given in section 6. Conclusions are given in section 7.

II. Learning Automata

The learning process of the *LA* is described as follows. The *LA* randomly selects an action from its action set and then performs it on the environment. The environment then evaluates the chosen action and responds with a reinforcement signal (reward or penalty) to the *LA*. According to the reinforcement signal of the environment to the selected action, the *LA* updates its action probability vector and then the learning process is repeated. The updating algorithm for the action probability vector is called the learning algorithm. The aim of the learning algorithm of the *LA* is to find an appropriate action from the set of actions so that the average reward received from the environment is maximized. The *LAs* can be classified into two classes, fixed and variable structure *LAs* [18], [19]. Variable structure *LAs* which is used in this paper is represented by sextuple $\langle \underline{\beta}, \underline{\phi}, \underline{\alpha}, \underline{P}, G, T \rangle$, where $\underline{\beta}$ a set of inputs actions (called response or reinforcement signal), $\underline{\phi}$ is a set of internal states, $\underline{\alpha}$ a set of outputs, P denotes the state probability vector, G is the output mapping, and T is learning algorithm. The learning algorithm is used to modify the probability vector.

$$\begin{aligned} p_i(k+1) &= p_i(k) + a\big(1 - p_i(k)\big) \\ p_j(k+1) &= p_j(k) - a p_j(k), \qquad \forall j \neq i \end{aligned} \tag{1}$$

$$p_i(k+1) = (1-b)p_i(k) \tag{2}$$
$$p_j(k+1) = \frac{b}{r-1} + (1-b)p_j(k), \qquad \forall j \neq i$$

Let α_i be the action chosen at step k as a sample realization from distribution *P(k)*. In a linear learning algorithm, equation for updating probability vector *P(k)* is defined by (1) for a favorable response (β=1), and (2) for an unfavorable response (β=0). The probability of getting unfavorable response by action α_i is denoted by $c_i = Pr[\beta = 0 \mid \alpha_i]$. Note that, the probabilities of getting unfavorable response or favorable response are unknown for the LA. Two parameters *a* and *b* represent reward and penalty parameters, respectively. The parameter *a* (*b*) determines the amount of increase (decreases) of the action probabilities. *r* denotes the number of actions that can be taken by the *LA*. If a=b, the above learning algorithm is called linear reward penalty (L_{RP}); if a>>b the learning algorithm is called linear reward□ε penalty ($L_{R\varepsilon P}$); and finally if b=0, it is called linear reward inaction (L_{RI}) algorithm. Learning automata have found applications in many areas such as sensor networks [4], [11], stochastic graphs [20], [21], peer-to-peer networks[22]–[27], channel assignment[28], mobile cloud computing[29], motion estimation [30], and cognitive networks[31] to mention a few.

III. Closed Asynchronous Dynamic Cellular Learning Automaton (CADCLA)

In this section, at first, a definition of *CADCLAs* is given, then its updating process is explained, and finally to study the updating process two metrics are introduced.

A. The Definition of the CADCLAs

A *CADCLA* is a network of cells whose structure changes with time. This model can be formally defined by a 8-tuple $CADCLA = (G, A, N, \Phi, \Psi, F_1, F_2, F_3)$, where:

- $G = (V, E)$ is an undirected graph which determines the structure of *CADCLA* where $V = \{cell_1, cell_2, \dots, cell_n\}$ is the set of vertices and *E* is the set of edges.
- $A = \{LA_1, LA_2, \dots, LA_n\}$ is a set of *LAs* each of which is assigned to one cell of *CADCLA*. The set of actions of automaton for a cell is the same as the set of states for that cell.
- $N = \{N_1, N_2, \dots, N_n\}$where $N_i = \{cell_j \in V \mid dist(cell_i, cell_j) < \theta_i\}$ where θ_i is the neighborhood radius of $cell_i$ and $dist(cell_i, cell_j)$ is the length of the shortest path between $cell_i$ and $cell_j$ in G. N_i^1 determines the immediate neighbors of $cell_i$ which constitute its **local environment**.
- $\Psi = \{\Psi_1, \Psi_2, \dots, \Psi_n\}$ where $\Psi_i = \{(j, X_j) \mid cell_j \in N_i\}$ denotes the attribute of $cell_j$ where $X_j \subseteq \{x_1, x_2, \dots, x_s\}$. $\{x_1, x_2, \dots, x_s\}$ is the set of allowable attributes. Ψ_i^1 determines the attribute of $cell_i$ when $\theta_i = 1$.
- $\Phi = \{\Phi_1, \Phi_2, \dots, \Phi_n\}$ where $\Phi_i = \{(j, \alpha_l) \mid cell_j \in N_i \text{ and action } \alpha_l \text{ has been chosen by } LA_j\}$ denotes the state of $cell_i$. Φ_i^1 determines the state of $cell_i$ when $\theta_i = 1$.
- $F_1\colon (\underline{\Psi}) \rightarrow (\underline{\zeta})$is the restructuring function. In each cell, the restructuring function computes the restructuring signal based on the attributes of the cell and its neighboring cells. For example, in $cell_i$, the restructuring function takes $< \Psi_i >$ and then returns a value from the closed interval [0,1]

for ζ_i^1 which is the restructuring signal of *cell_i*. we define $\zeta_i = \{(j, \zeta_j^1) | cell_j \in N_i\}$ to be the set of restructuring signals of neighbors of *cell*$_i$. In a cell, depending the application the value of the restructuring signal determines whether the neighbors of that cell should be changed or not. If the ζ_i^1 is equal to zero (one) this means that the neighbors of cell$_i$ are appropriate (not appropriate).

- $F_2: \left(\underline{N}, \underline{\Psi}, \underline{\zeta}\right) \rightarrow (\underline{N}^1)$ is the structure updating rule. In each cell, the structure updating rule finds the immediate neighbors of the cell based on the restructuring signal computed by the cell, the attributes of the neighbors of the cell, and the neighbors of the cell. For example, in *cell*$_i$, structure updating rule takes $< N_i, \Psi_i, \zeta_i >$ and returns $< N_i^1 >$.

- $F_3: (\underline{\Phi}, \underline{\Psi}) \rightarrow \left(\underline{\beta}\right)$ is the local rule of *CADCLA*. In each cell, the local rule computes the reinforcement signal for the learning automata of that cell based on the states and the attributes of that cell and its neighboring cells. For example, in *cell*$_i$, local rule takes $< \Phi_i, \Psi_i >$ and then computes the reinforcement signal $< \beta_i >$ for the learning automata of *cell*$_i$.

B. *The updating process of CADCLAs*

The updating process of *CADCLAs* is described as follow. Note that, the order by which the cells of *CLA* will be activated is application dependent. Upon the activation of a cell, the cell performs a process which has three phases: ***preparation***, ***structure updating***, and ***state updating***. These three phases are described below.

1- Preparation phase: In this phase, a cell performs the following steps.

Step.1: The cell determines its neighborhood radius and attribute.

Step.2: The cell and its neighboring cells compute their *restructuring signals* using the *restructuring function (F_1)*.

2- Structure updating phase: In this phase, a cell performs the following steps.

Step.1: The neighborhood structure of the cell is updated using the structure updating rule (F_2) if the value of the restructuring signal of that cell is 1.

Step.2: The cell goes to the **state updating phase**

3- State updating phase: In this phase, a cell performs the following steps

Step.1: Each learning automaton of the cell selects one of its actions. The set of actions selected by the set of learning automata in the cell determines the new state for that cell.

Step.2: The local rule (F_3) is applied and a reinforcement signal is generated according to which the action probability vectors of the learning automata of the cell are updated.

C. *The performance metrics of CADCLAs*

The performance of *CADCLAs* will be studied using two metrics: ***entropy***, and ***potential energy*** which are defined as below.

Entropy: The *entropy* of the CLA at iteration *t* is defined by equation (3) given below

$$H(t) = -\sum_{k=1}^{n}\sum_{l=1}^{r_k} p_{kl}(t) \times \ln(p_{kl}(t)) \qquad (3)$$

In the equation (3), *n* is the number of *LAs* of the *CADCLA*. r_k is the number of actions of the LA_k. p_{kl}(t) is the probability of selecting action α_l of the LA_k at iteration *t* of the *CADCLA*. Entropy of the *CADCLA* can be used to study the changes occur in the states of the cells of *CADCLA*. Higher values of *H(t)* mean higher rates of changes in the actions selected by *LAs* residing in the cells of the *CADCLA* [4], [11].

Potential energy: The *potential energy* of the CLA is defined by equation (4) given below

$$\mathrm{A(t)}=\sum_{i=1}^{n}\zeta_i^1(\mathrm{t}) \qquad (4)$$

where $\zeta_i^1(\mathrm{t})$ is the restructuring signal of $cell_i$ at iteration *t*. *Potential energy* can be used to study the changes in the structure of *CLA* as it interacts with the environment. Higher value of *A(t)* indicates higher disorder in the structure of *CLA*.

IV. PRELIMINARIES

In order to study the expediency of *CADCLAs*, required definitions are given in this section.

Definition 1. The configuration of the *CADCLA* at iteration t, is defined as $S(t) = \langle \underline{N}(t), \underline{\mathrm{P}}(t), \underline{\Phi}(t)\rangle$ where

- $\underline{\mathrm{N}}(t) = (\mathrm{N}_1(t), \mathrm{N}_2(t), \dots, \mathrm{N}_n(t))^{\mathrm{T}}$
- $\underline{P}(t) = (\mathrm{P}_1(t), \mathrm{P}_2(t), \dots, \mathrm{P}_\mathrm{n}(t))^{\mathrm{T}}$ where $\mathrm{P_i}(t) = (p_{i1}(t), p_{i2}(t), \dots, p_{ir}(t))^{\mathrm{T}}$ in which $p_{ij}(t)$ is the probability of selecting action α_j of the learning automaton LA_i at iteration *t.* Each learning automaton has r actions.
- $\underline{\Phi}(t) = (\Phi_1(t), \Phi_2(t), \dots, \Phi_n(t))^{\mathrm{T}}$

The initial configuration of the *CADCLA* denoted by $S(0) = \langle \underline{N}(0), \underline{\mathrm{P}}(0), \underline{\Phi}(0)\rangle$. As it was previously mentioned, upon activating the cells, a process takes the configuration, reinforcement signals, restructuring signals, and then updates the configuration of the *CADCLA*. The evolution of *CADCLA* can be described by sequence $\{S(t)\}_{t\geq 0}$ which $< S(t+1) >= \underline{z}(S(t), \underline{\beta}(\mathrm{t}), \underline{\zeta}(\mathrm{t}))$.

Definition 2. A *CADCLA* is said to be expedient with respect to the cell c_i if the following inequality holds:

$$\lim_{\mathrm{t}\to\infty} E\left[D_i^{\beta}(S(t))\right] > \lim_{\mathrm{t}\to\infty} D_i^{pc}(\mathrm{S^{pc}}(t)) \qquad (5)$$

Before we define the elements of the above equitation, we need to define the following items.

- *LG(i,j,k,t)* takes i (index of a cell), j (index of an action), k (index of an attribute), and t (iteration number) and then return $< \Phi_\mathrm{i}^*, \Psi_\mathrm{i}^* >$ where Φ_i^* is a version of $\Phi_i(\mathrm{t})$ which its item (i,-) is replaced with (i, α_j) and Ψ_i^* is a version of $\Psi_i(\mathrm{t})$ which its item (i,-) is replaced with (i, b_k). note that (i,-) refer to every item which its first element is equal to i.

- *LPG (i,j,k,t)* returns the probability of appearing a set which is returned by *LG(i,j,k,t)* in iteration t.
- $q_{\mathrm{ijk}}^{\beta}(S(t))$ denotes the reward probability of action α_j of learning automaton *LA_i* when the index of attribute of *cell_i* is k . $q_{\mathrm{ijk}}^{\beta}(S(t))$ is defined by equation (6) as given below

$$q_{\mathrm{ijk}}^{\beta}(S(t)) = E[\beta_i = 1|(i, \alpha_j) \epsilon\, \Phi_i(t), (i, b_\mathrm{k}) \in \Psi_i(\mathrm{t})] = \sum_j \sum_k \left(\mathrm{LPG(i, j, k, t)} \times F_4^i(\mathrm{LG(i, j, k, t)})\right) \tag{6}$$

- $d_{ij}^{\beta}(S(t))$ denotes the reward probability of action α_j of learning automaton *LA_i*. $d_{ij}^{\beta}(S(t))$ is defined by equation (7) as given below

$$d_{ij}^{\beta}(S(t)) = \sum_k q_{ijk}^{\beta}(S(t)) \tag{7}$$

Now we define the $D_i^{\beta}(S(t))$ and $D_i^{pc}(\mathrm{S}^{\mathrm{pc}}(t))$

$$D_i^{\beta}(S(t)) = E[\beta_i = 1| < \Phi_i(\mathrm{t}), \Psi_i(\mathrm{t}) >] = \sum_j \left(p_{ij}(t) \times d_{ij}^{\beta}(S(t))\right) \tag{8}$$

$D_i^{pc}(\mathrm{S}^{\mathrm{pc}}(t))$ where $\mathrm{S}^{\mathrm{pc}}(t) = \langle \underline{\mathrm{P}}^{\mathrm{pc}}(t), \underline{\mathrm{N}}^{\mathrm{pc}}(t), \underline{\Phi}^{\mathrm{pc}}(t) \rangle$ denotes the average reward received by *cell_i* of in pure-chance *CADCLA*. A pure-chance *CADCLA* is a *CADCLA* in which, each *LA* is replaced with a pure-chance automaton. A pure-chance automaton is an automaton that always selects each of its actions with equal probabilities [18]. The probability vectors of pure-chance automata of the *CADCLA* remains unchanged during the iterations of *CADCLA*. Note that, $LPG(i, j, k, t)^{pc}$ is the probability of appearing the set $LG(i, j, k, t)^{pc}$ in pure-chance *CADCLA*.

Definition 3. A *CADCLA* is said to be *expedient,* if it is *expedient* which respect to each cell.

Definition 4. A *CADCLA* is said to be ε-optimal with respect to cells, if for each *cell_i* in which $d_{il}^{\beta}(S) > d_{ik}^{\beta}(S)$ where $l \neq k$, we have the following inequality.

$$\lim_{t \to \infty} \inf p_{il}(t) > 1 - \varepsilon \quad \text{w.p.1} \tag{9}$$

Definition 5. The *structure updating rule* is called *potential-decreasing*, if applying the *structure updating rule* in each cell leading to strictly decreasing the function *potential energy*.

Proposition 1. if the *structure updating rule* is *potential-decreasing* and $t \to \infty$, then there is an iteration $t' < t$ which $A(t) = 0$.

Proof. In each iteration, utilizing *potential-decreasing structure updating rule* leading to strictly decreasing the value of potential energy over time. Note that, the minimum value of the potential energy is zero.

Therefore, there is an iteration t' which the potential energy remains zero.

V. Expediency of *CADCLAs*

The expediency of the *CADCLA* has been studied in this section. In this section, the environment under which LA_{ij} in the *CLA* is operating can be modeled as follows,

1. There is function a $f_{ij}^{\beta}(\mathrm{P_i})$ which $f_{ij}^{\beta}(\mathrm{P_i}(t)) = d_{ij}^{\beta}(S(\mathrm{t}))$ and it is continuous in p_{ik} (j, k=1,2,…,r).
2. $\frac{\partial f_{ij}^{\beta}(\mathrm{P_i})}{\partial p_{ij}} < 0$
3. $\frac{\partial f_{ik}^{\beta}(\mathrm{P_i})}{\partial p_{ij}} \gg \frac{\partial f_{ij}^{\beta}(\mathrm{P_i})}{\partial p_{ij}}$ for $k \neq j$
4. $f_{ij}^{\beta}(.)$ is continuously differentiable in all its arguments.
5. $f_{ij}^{\beta}(\mathrm{P_i})$ and $\left(\frac{\partial f_{ij}^{\beta}(\mathrm{P_i})}{\partial p_{ij}}\right)$ are Lipschitzian functions of all their arguments.
6. In every *cell$_i$*, there is an action α_l which $f_{il}^{\beta}(\mathrm{P_i}) > f_{ik}^{\beta}(\mathrm{P_i})$ where $l \neq k$.
7. $a \to 0$ where a is the reward parameter.

A. *Expediency of CADCLA with L_{RP} Learning Algorithm for the LAs*

In this section, we suggest a set of sufficient conditions under which the *CADCLA* with L_{RP} learning algorithm for the *LAs* is expedient.

Lemma 1. If the non-stationary environment of a *LA* with $\mathrm{L_{RP}}$ Learning Algorithm satisfies the following conditions

- c_i $(p_1,p_2,...,p_r)$ is continuous in p_j $(i,j$=1,2,…,r) if p_j(t)= p_j. c_i $(p_1,p_2,...,p_r)$ denotes the penalty probability of choosing action α_i and p_i denotes the probability of choosing action α_i.
- $\frac{\partial c_i(p)}{\partial p_i} > 0$
- $\frac{\partial c_j}{\partial p_i} \ll \frac{\partial c_i}{\partial p_i}$ for $j \neq i$
- $c_i(.)$ is continuously differentiable in all its arguments.
- $a \to 0$ where a is the reward parameter.

Then, the process $\{p_i(t)\}_{t\geq 0}$ is Markovian and ergodic, which satisfy the following equation.

$$\lim_{t\to\infty} E[p_i(t) - p_i^*]^2 = 0 \tag{10}$$

Where p^* is the probability vector of the LA which satisfy the following equality.

$$p_1^* \times c_1(p^*) = p_2^* \times c_2(p^*) = \ldots = p_r^* \times c_r(p^*) \tag{11}$$

Proof. The proof is given in section 7 of [19].

Theorem 1. In a *CADCLA,* if the learning algorithms of *LAs* are L_{RP}, then regardless to the initial configuration, *restructuring function* and *structure updating rule*, we have the following equation.

$$\lim_{t\to\infty}\left(\underline{\mathrm{P}}(t)\right) = \underline{\mathrm{P}}^* \tag{12}$$

Proof. Since all conditions mentioned in lemma 1 for the environment of the *LA* are also mentioned in the environment of the *LA* of each cell, the environment of a *LA* of the *CADCLA* is similar to the environment of the *LA* in the Lemma 1. As result of lemma 1, the *LA* of *cell$_i$* tries to find probability vector $\mathrm{P}_\mathrm{i}^* = (p_{\mathrm{i}1}^*, p_{\mathrm{i}2}^*, \ldots, p_{\mathrm{ir}}^*)^{\mathrm{T}}$ which satisfy the equation (13). This phenomenon occurs in all cells and therefore the probability vectors of all *LAs* approach to $\underline{\mathrm{P}}^*$ and the proof is completed.

$$p_{i1}^* \times \left(1 - f_{i1}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = p_{i2}^* \times \left(1 - f_{i2}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = \ldots = p_{ir}^* \times \left(1 - f_{ir}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) \tag{13}$$

Lemma 2. In a *CADCLA,* if the learning algorithms of *LAs* are L_{RP}, then regardless to the initial configuration, *restructuring function* and *structure updating rule* , we have $1 - (\mathrm{r} \times w_i) = \sum_{\mathrm{j}=1}^{\mathrm{r}} p_{ij}^* \times f_{ij}^{\beta}(\mathrm{P}_\mathrm{i}^*)$ where $w_i = p_{ij}^* \times (1 - f_{ij}^{\beta}(\mathrm{P}_\mathrm{i}^*))$ which $\mathrm{j} \in \{1 \ldots \mathrm{r}\}$.

Proof. all conditions mentioned in lemma.1 are satisfied for every cell of the *CADCLA*. According to the results of lemma 1, equation (14) is correct for *cell$_i$*.

$$p_{i1}^* \times \left(1 - f_{i1}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = p_{i2}^* \times \left(1 - f_{i2}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = \ldots = p_{ir}^* \times \left(1 - f_{ir}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = w_i \tag{14}$$

$$p_{i1}^* \times \left(1 - f_{i1}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) + p_{i2}^* \times \left(1 - f_{i2}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) + \ldots + p_{ir}^* \times \left(1 - f_{ir}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) = r \times w_i \tag{15}$$

Equation 15 is obtained by using equation 14.

$$(p_{i1}^* + p_{i2}^* + \ldots + p_{ir}^*) - (p_{i1}^* \times f_{i1}^{\beta}(\mathrm{P}_\mathrm{i}^*) + p_{i2}^* \times f_{i2}^{\beta}(\mathrm{P}_\mathrm{i}^*) + \ldots + p_{ir}^* \times f_{ir}^{\beta}(\mathrm{P}_\mathrm{i}^*) = \mathrm{r} \times w_i \tag{16}$$

$$1 - \left(\sum_{\mathrm{j}=1}^{\mathrm{r}} \left(p_{ij}^* \times f_{ij}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right)\right) = \mathrm{r} \times w_i \tag{17}$$

$$1 - (\mathrm{r} \times w_i) = \sum_{\mathrm{j}=1}^{\mathrm{r}} \left(p_{ij}^* \times f_{ij}^{\beta}(\mathrm{P}_\mathrm{i}^*)\right) \tag{18}$$

Equation 18 is obtained by simplifying equation 15 and the proof is complete.

Theorem 2. In a *CADCLA,* if the learning algorithms of *LAs* are L_{RP}, and for every *cell*$_i$ $w_i < \frac{1}{r} - \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})}{r^2}$, then the *CADCLA* is expedient regardless to its initial configuration, *restructuring function* and *structure updating rule*.

Proof. In order to prove that an *CADCLA* is expedient with respect to cells, we need to show that the inequality $\lim_{\mathrm{t}\to\infty} E\left[D_i^{\beta}\left(S(t)\right)\right] > \lim_{\mathrm{t}\to\infty} D_i^{pc}\left(\mathrm{S}^{\mathrm{pc}}(t)\right)$ is hold for every *cell*$_i$. By expanding both sides of this inequality, we have the following inequality.

$$\lim_{t\to\infty} E\left[\sum_j\left(p_{ij}(t) \times f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right)\right] > \lim_{t\to\infty} \sum_j \left(p_{ij}^{pc}(t) \times f_{ij}^{\beta}\left(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}(t)\right)\right) \tag{19}$$

Again, by expanding both sides, we have the following inequality. Since the probability vector of a pure-chance automaton do not change over time, the index t of $\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}(t)$ is omitted in equation (20).

$$\lim_{t\to\infty} E\left[p_{i1}(t) \times f_{i1}^{\beta}(\mathrm{P}_{\mathrm{i}}(t)) + p_{i2}(t) \times f_{i2}^{\beta}(\mathrm{P}_{\mathrm{i}}(t)) + \cdots + p_{ir}(t) \times f_{ir}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right] > \lim_{t\to\infty}\left(p_{i1}^{pc}(t) \times f_{i1}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}) + p_{i2}^{pc}(t) \times f_{i2}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}) + \cdots + p_{ir}^{pc}(t) \times f_{ir}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})\right) \tag{20}$$

Now, by applying expectation, we have the following inequality.

$$\lim_{t\to\infty} \left(E\left[p_{i1}(t) \times f_{i1}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right] + \mathrm{E}\left[p_{i2}(t) \times f_{i2}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right] + \cdots + E\left[p_{ir}(t) \times f_{ir}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right]\right) > \lim_{t\to\infty}\left(p_{i1}^{pc}(t) \times f_{i1}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}) + p_{i2}^{pc}(t) \times f_{i2}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}}) + \cdots + p_{ir}^{pc}(t) \times f_{ir}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})\right) \tag{21}$$

After replacing $p_{ij}^{pc}(t)$ with $\frac{1}{r}$ in the right hand side, equation 21 changes to equation 22.

$$\lim_{t\to\infty} \left(E\left[p_{i1}(t) \times f_{i1}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right] + \mathrm{E}\left[p_{i2}(t) \times f_{i2}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right] + \cdots + E\left[p_{ir}(t) \times f_{ir}^{\beta}(\mathrm{P}_{\mathrm{i}}(t))\right]\right) > \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})}{r} \tag{22}$$

According to the result of lemma 2, equation (22) changes to equation (23).

$$\sum_{\mathrm{j}=1}^{\mathrm{r}}\left(p_{ij}^{*} \times\ f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{*})\right) > \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})}{r} \tag{23}$$

By substituting equation (18) in equation (22) we have the following.

$$1 -\ (\mathrm{r} \times w_i) > \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})}{r} \tag{24}$$

$$w_i < \frac{1}{r} - \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P}_{\mathrm{i}}^{\mathrm{pc}})}{r^2} \tag{25}$$

And the proof is complete.

Proposition 2. If the conditions mentioned in Theorem 2 are satisfied, then by proper choice of parameters of the *LAs*, the entropy of the *CADCLA* approaches to a constant value h^*.

Proof. Note that, H(t) refers to the entropy of the *CADCLA* in iteration t. By expanding $\lim_{t\to\infty}(\mathrm{H(t)})$ we have $\lim_{t\to\infty}\left(-\sum_{k=1}^{n}\sum_{l=1}^{r_k}[p_{kl}(t)\times\ln(p_{kl}(t))]\right)$. According to the result of Theorem 1, we have $\lim_{t\to\infty}(\underline{\mathrm{P}}(t))=\underline{\mathrm{P}}^*$. Therefore, H(t) approaches to $-\sum_{k=1}^{n}\sum_{l=1}^{r_k}[p_{kl}^*\times\ln(p_{kl}^*)]$ which is a constant value and the proof is complete.

Proposition 3. If the conditions mentioned in Theorem 2 are satisfied, and *structure updating rule* is *potential-decreasing*, then regardless to its initial configuration, the configuration of the *CADCLA* approaches to configurations in which $\lim_{t\to\infty}(A(t))=0$ and $\lim_{t\to\infty}(H(t))=\square^*$ where $h^*$is a constant value.

Proof. According to the results of proposition 1 and proposition 2, the proof is straightforward.

B. Expediency of CADCLA with $L_{R\varepsilon P}$ Learning Algorithm for the LAs

In this section, we suggest a set of sufficient conditions under which the *CADCLA* with $L_{R\varepsilon P}$ learning algorithm for the *LAs* is expedient.

Lemma 3. If the non-stationary environment of a *LA* with $L_{R\varepsilon P}$ learning algorithm satisfies the following conditions

- $c_i(p_1,p_2,\ldots,p_r)$ is continuous in p_j (i,j=1,2,…,r).
- $\frac{\partial c_i(p)}{\partial p_i}>0$
- $\frac{\partial c_j}{\partial p_i}\ll\frac{\partial c_i}{\partial p_i}$ for $j\neq i$
- $c_i(.)$ is continuously differentiable in all its arguments.
- $c_i(.)$ and $\left(\frac{\partial c_i}{\partial p_i}\right)$ are Lipschitzian functions of all their arguments.
- There exist an action α_l which $c_k(p)>c_l(p)$ for $k\neq l$

Then, by proper choice of parameters in the *LAs* and for any given ε >0, the process $\{p(t)\}_{t\geq 0}$ is Markovian and ergodic, which satisfy the following inequality.

$$\lim_{t\to\infty}\inf p_l(t)>1-\varepsilon \qquad \text{w.p.1} \tag{26}$$

Where $p_l(t)$ denotes the probability of selecting action α_l and $p(t)$ denotes the probability vector of the *LA* in iteration t.

Proof. The proof is given in section 7 of [19].

Theorem 3. For any given ε>0 and by proper choice of parameters in the *LAs*, if the learning algorithms of *LAs* are $L_{R\varepsilon P}$, then regardless to its initial configuration *restructuring function* and *structure updating rule*, the *CADCLA* is ε-optimal with respect to cells.

Proof. All conditions mentioned in lemma 3 for the environment of the *LA* are also mentioned in the environment of the *LA* of each cell. Then the environment of a *LA* of the *CADCLA* is similar to the environment of the *LA* in the Lemma 3. According to the results of lemma 3, the probability of selecting action α_l of *LA*$_i$ (*cell*$_i$) which α_l has the highest reward probability among the actions in the action set of *LA*$_i$ approaches to a value higher than $1-\varepsilon$. This phenomenon occurs in all cells. In other word, the *CADCLA* is ε-optimal with respect to cells according to definition 4 and the proof is complete.

Theorem 4. If the learning algorithms of *LAs* are $L_{R\varepsilon P}$, and for every cell such as *cell*$_i$ we have $\lim_{t\to\infty} E\left[p_{il}(t)\times f_{il}^{\beta}(\mathrm{P_i}(t))\right] > \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P_i^{pc}})}{r} - \lim_{t\to\infty} E\left[\sum_{j\neq l} p_{ij}(t)\times f_{ij}^{\beta}\left(\mathrm{P_i}(t)\right)\right]$, then the *CADCLA* is expedient with respect to cells regardless to its initial configuration, *restructuring function* and *structure updating rule*.

Proof. According to definition 2, in order to prove this theorem, we need to show that the inequality $\lim_{\mathrm{t}\to\infty} E\left[D_i^{\beta}\left(S(t)\right)\right] > \lim_{t\to\infty} D_i^{pc}\left(\mathrm{S^{pc}}(t)\right)$ holds for every *cell*$_i$. By expanding both sides of this inequality, we have the following inequality.

$$\lim_{\mathrm{t}\to\infty} E\left[D_i^{\beta}\left(S(t)\right)\right] > \lim_{t\to\infty} \sum_j\left(p_{ij}^{pc}(t)\times f_{ij}^{\beta}(\mathrm{P_i^{pc}}(t))\right) \tag{27}$$

Note that, the right hand side of (27) is not function of t because the probability vector of the pure-chance automata does not change over time. By expanding the left hand side and replacing $p_{ij}^{pc}(t)$ in right hand side of (27) with $\frac{1}{r}$, we have the following inequality.

$$\lim_{t\to\infty} E\left[p_{il}(t)\times f_{il}^{\beta}(\mathrm{P_i}(t))\right] > \frac{\sum_{j=1}^{r} f_{ij}^{\beta}(\mathrm{P_i^{pc}})}{r} - \lim_{t\to\infty} E\left[\sum_{j\neq l} p_{ij}(t)\times f_{ij}^{\beta}\left(\mathrm{P_i}(t)\right)\right] \tag{28}$$

And the proof is complete.

Proposition 4. if conditions mentioned in Theorem 4 are satisfied and the parameters of the *LAs* are chosen properly, then $\lim_{t\to\infty}\left(\mathrm{H(t)}\right) = \square^{\varepsilon}$ where $\square^{\varepsilon}$is a value which depends on ε.

Proof. by expanding $\lim_{t\to\infty}\left(\mathrm{H(t)}\right)$, we have the following equations.

$$\lim_{t\to\infty}\left(-\sum_{k=1}^{n}\sum_{l=1}^{r_k}[p_{kl}(t)\times \ln(p_{kl}(t))]\right) \tag{29}$$

$$\lim_{t\to\infty}\left(-\sum_{k=1}^{n}\left[p_{k1}(t)\times\ln(p_{k1}(t))+p_{k2}(t)\times\ln(p_{k2}(t))+\cdots+p_{kr_k}(t)\times\ln\left(p_{kr_k}(t)\right)\right]\right) \tag{30}$$

As a result of Theorem 3, by proper choose of parameters in the *LAs*, we have $\lim_{t\to\infty}\inf p_{il}(t) > 1-\varepsilon$ with probability 1 for action α_l of *LA*$_i$. Note that, if the value of $p_{il}(t)$ approaches to a value higher than $1-\varepsilon$, the summation of probabilities of selecting other actions of *LA*$_i$ approaches to a value lower than ε. This phenomenon occurs in all *LAs* of the *CADCLA* and therefore, $\lim_{t\to\infty}(\mathrm{H}(\mathrm{t}))$ approaches to $\square^{\varepsilon}$which $\square^{\varepsilon}$ is a value which depends on ε.

Proposition 5. If conditions mentioned in Theorem 4 are satisfied, and *structure updating rule* is *potential-decreasing*, then regardless to its initial configuration, the configuration of the *CADCLA* approaches to a configuration in which $\lim_{t\to\infty}(H(t)) = \square^{\varepsilon}$ and $\lim_{t\to\infty}(A(t)) = 0$.

Proof. According to the results of proposition 1 and proposition 4, the proof is straightforward.

VI. Simulation

In this section, the simulation environment is described and then two experiments are given to support the theoretical results given in section 5.

A. Simulation Environment and Setup

Figure 1 is used to construct the *CADCLA*. In this *CADCLA*, five cells are used with the following descriptions. Each cell is equipped with one *LA*. Each *LA* has two actions: "ON" and "OFF". The values of $\Psi_1^1(0)$, $\Psi_2^1(0), \Psi_3^1(0), \Psi_4^1(0), \Psi_5^1(0)$ are set to "RED", "BLUE", "RED", "BLUE", "RED" respectively. The values of $\Phi_1^1(0)$, $\Phi_2^1(0), \Phi_3^1(0), \Phi_4^1(0), \Phi_5^1(0)$ are set to "ON", "ON", "ON", "ON" , "ON" respectively. In this section, an activation sequence determines the order under which the cells are activated. The mechanism used for generating the activation sequence is described as follows. A activation sequence with order k is composed of concatenation of k random permutation of indices of the cells. The algorithm reported in [32] known as the Knuth shuffle is used to generate the random permutation of indices. The order of activation sequence is equal to 2000.

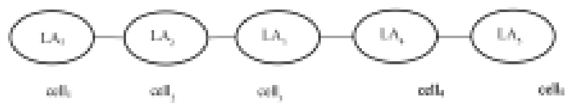

Figure 1. Initial cellular structure of the *CADCLA*

To complete the description of the simulation, we need to describe the routine executed by a cell after activation. Upon activation of a cell, the cell performs the preparation phase. During the preparation phase, the *restructuring signal* of the cell is computed. In this phase, a cell using a function called similarity function calculates the portion of its neighbors which they have similar attributes with that cell. If the value of the similarity function is lower than a threshold 0.5, the restructuring function returns 1 as restructuring signal and 0 otherwise. At the end of preparation phase, the cell goes to structure updating phase. In this phase, the cell decides whether or not to change its neighbors. If the value of the restructuring signal is equal to 1, the cell randomly selects one of its neighbors and then swaps its neighbors with that neighbors in order to increase the number of its similar neighbors. At the end of structure updating phase, the cell goes to the state updating phase. During the state updating phase, the learning automaton of the cell selects one of its actions. The action selected by the learning automaton in the cell determines the new state for that cell. The response of the environment to the selected action is generated by (33). In (33), matrix $D(t) = \left[d_{ij}(t)\right]_{5\times 2}$ Denotes the reward probabilities matrix in which $d_{ij}(t)$ denoted the reward probability of learning automaton LA_i and action α_j in iteration t.

$$D(t) = \begin{bmatrix} \left(1 - \frac{p_{11}(t)}{3}\right) & \frac{p_{11}(t)}{3} \\ \frac{p_{22}(t)}{3} & \left(1 - \frac{p_{22}(t)}{3}\right) \\ \frac{p_{32}(t)}{3} & \left(1 - \frac{p_{32}(t)}{3}\right) \\ \left(1 - \frac{p_{41}(t)}{3}\right) & \frac{p_{41}(t)}{3} \\ \left(1 - \frac{p_{51}(t)}{3}\right) & \frac{p_{51}(t)}{3} \end{bmatrix} \tag{33}$$

B. *Experiment 1*

This experiment is conducted to confirm that if the condition mentioned in proposition 5 are satisfied, then we have $\lim_{t\to\infty}(A(t)) = 0$ and $\lim_{t\to\infty}(H(t)) = h^{\varepsilon}$. In this experiment, the learning algorithms of *LAs* are $L_{R\varepsilon P}$. The reward parameter and penalty parameter of the *LAs* are set to 0.001 and 0.00001 respectively. The conditions mentioned for proposition 5 are satisfied in this experiment. The results show that the value of *Entropy* approaches to a very small value and the value of *Potential Energy* approach to zero. Therefore, the configuration of the *CADCLA* approaches to a fixed configuration.

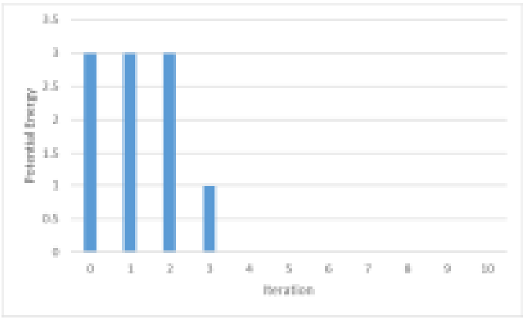

Figure 2. the Potential Energy for *CADCLA* with $L_{R\varepsilon P}$ learning algorithm for the *LAs*

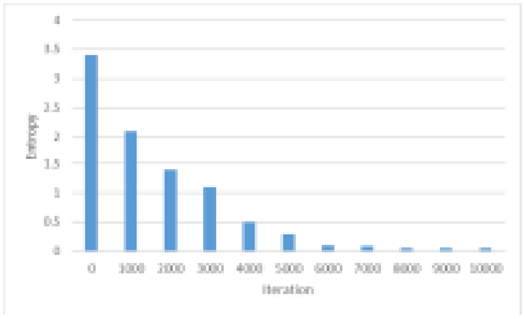

Figure 3. the Entropy for *CADCLA* with $L_{R\varepsilon P}$ learning algorithm for the *LAs*

C. *Experiment 2*

This experiment is conducted to confirm that if the condition mentioned in proposition 3 are satisfied, then we have $\lim_{t\to\infty}(A(t)) = 0$ and $\lim_{t\to\infty}(H(t)) = \square^*$ where $h^*$is a constant value. In this experiment, the learning algorithms of *LAs* are L_{RP}. In this experiment, both the reward and penalty parameters of the *LAs* are set to 0.001. The conditions mentioned for proposition 3 are satisfied in this experiment. The results show that the values of *Potential Energy* of *CADCLA* approach to zero. Therefore, the cellular structure approaches to a fixed structure.

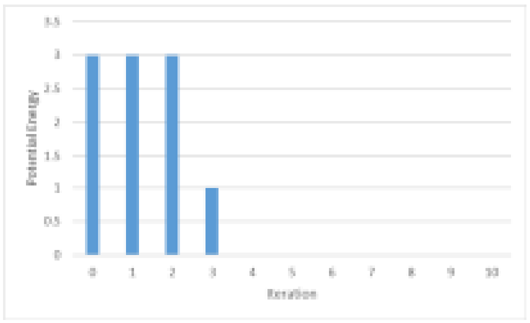

Figure 4. the Potential Energy for *CADCLA* with L$_{RP}$ learning algorithm for the *LAs*

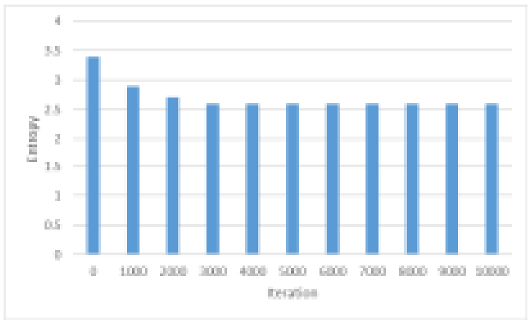

Figure 5. the Entropy for *CADCLA* with L$_{RP}$ learning algorithm for the *LAs*

VII. Conclusion

In this paper, we suggested sufficient conditions under which a *CADCLA* can be expedient. This study is applicable for both L_{RP} and $L_{R\varepsilon P}$ learning algorithms. A numerical example was given for supporting the theoretical results. It should be noted that, the mentioned conditions for the *CADCLAs* are not application dependent. Satisfying these conditions guarantee the expediency of the *CADCLA* in every *CADCLA* based algorithm. Note that, *CLAs* have been used in several applications such as computer networks, image processing, and social networks. *CADCLAs* are able to support different forms of dynamicity in *CLAs*. Therefore, they can be used in several applications. As future work, we plan to study the expediency of the *CADCLA* with L_{RI} learning algorithm for the *LAs*. In addition, we plan to define new metrics to evaluate the behavior of the *CADCLA*.

References

[1] S. Wolfram, "Theory and applications of cellular automata," *World Scientific Publication*, 1986.

[2] J. Kroc, P. M. A. Sloot, and A. Georgius Hoekstra, *Simulating Complex Systems by Cellular Automata*, Understanding Complex Systems. Springer, 2010.

[3] H. Beigy and M. R. Meybodi, "A mathematical framework for cellular learning automata," *Advances in Complex Systems*, vol. 3, no. 4, pp. 295–319, 2004.

[4] M. Esnaashari and M. Meybodi, "A cellular learning automata-based deployment strategy for mobile wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 71, no. 5, pp. 988–1001, 2011.

[5] M. Esnaashari and M. Meybodi, "A cellular learning automata based clustering algorithm for wireless sensor networks," *Sensor Letters*, vol. 6, no. 5, pp. 723–735, 2008.

[6] H. Beigy and M. R. Meybodi, "A self-organizing channel assignment algorithm: A cellular learning automata approach," *Intelligent Data Engineering and Automated Learning*, vol. 14, pp. 119–126, 2003.

[7] M. Asnaashari and M. R. Meybodi, "Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks," in *Proceedings of 15th Conference on Electrical Engineering*, Tehran, Iran, 2007, pp. 21–28.

[8] Y. Zhao, W. Jiang, S. Li, Y. Ma, G. Su, and X. Lin, "A cellular learning automata based algorithm for detecting community structure in complex networks," *Neurocomputing*, vol. 151, pp. 1216–1226, 2015.

[9] M. Vahidipour, M. R. Meybodi, and M. Esnaashari, "Adaptive Petri Net Based on Irregular Cellular Learning Automata and Its Application in Vertex Coloring Problem Systems with Unknown Parameters," *Applied Intelligence*, 2016.

[10] 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*, vol. 3, no. 2, pp. 83–98, 2006.

[11] M. Esnaashari and M. Meybodi, "Deployment of a Mobile Wireless Sensor Network with k-Coverage Constraint: A Cellular Learning Automata Approach," *Wireless Networks*, vol. 19, no. 5, pp. 945–968, 2013.

[12] A. M. Saghiri and M. R. Meybodi, "An Approach for Designing Cognitive Engines in Cognitive Peer-to-Peer Networks," *Journal of Network and Computer Applications*, vol. 70, pp. 17–40, 2016.

[13] H. Beigy and M. R. Meybodi, "Open synchronous cellular learning automata," *Advances in Complex Systems*, vol. 10, no. 4, pp. 527–556, 2007.

[14] H. Beigy and M. R. Meybodi, "Asynchronous cellular learning automata," *Automatica*, vol. 44, no. 5, pp. 1350–1357, 2008.

[15] H. Beigy and M. R. Meybodi, "Cellular learning automata with multiple learning automata in each cell and its applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 1, pp. 54–65, 2010.

[16] M. Esnaashari and M. R. Meybodi, "Irregular Cellular Learning Automata," *IEEE Transactions on Cybernetics*, no. 99, p. 1, 2014.

[17] M. Mozafari, M. E. Shiri, and H. Beigy, "A cooperative learning method based on cellular learning automata and its application in optimization problems," *Journal of Computational Science*, 2015.

[18] M. Thathachar and P. S. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Dordrecht, Netherlands: Kluwer Academic Publishers, 2004.

[19] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[20] A. R. Rezvanian and M. R. Meybodi, “Finding Maximum Clique in Stochastic Graphs Using Distributed Learning Automata,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 23, no. 1, pp. 1–31, 2015.

[21] A. R. Rezvanian and M. R. Meybodi, “Sampling Algorithms for Stochastic Graphs: A Learning Automata Approach,” *Knowledge Based Systems*, 2017.

[22] S. Gholami, M. R. Meybodi, and A. M. Saghiri, “A Learning Automata-Based Version of SG-1 Protocol for Super-Peer Selection in Peer-to-Peer Networks,” in *Proceedings of the 10th International Conference on Computing and Information Technology*, Angsana Laguna, Phuket, Thailand, 2014, pp. 189–201.

[23] M. Ghorbani, M. R. Meybodi, and A. M. Saghiri, “A new version of k-random walks algorithm in peer-to-peer networks utilizing learning automata,” in *5th Conference on Information and Knowledge Technology*, Shiraz,Iran, 2013, pp. 1–6.

[24] M. Ghorbani, M. R. Meybodi, and A. M. Saghiri, “A novel self-adaptive search algorithm for unstructured peer-to-peer networks utilizing learning automata,” in *3rd Joint Conference of AI & Robotics and 5th RoboCup Iran Open International Symposium*, Qazvin,Iran, 2013, pp. 1–6.

[25] A. M. Saghiri and M. R. Meybodi, “A distributed adaptive landmark clustering algorithm based on mOverlay and learning automata for topology mismatch problem in unstructured peer-to-peer networks,” *International Journal of Communication Systems*, 2015.

[26] A. M. Saghiri and M. R. Meybodi, “A Self-adaptive Algorithm for Topology Matching in Unstructured Peer-to-Peer Networks,” *Journal of Network and Systems Management*, 2015.

[27] A. M. Saghiri and M. R. Meybodi, “A Closed Asynchronous Dynamic Model of Cellular Learning Automata and its Application to Peer-to-Peer Networks,” *Genetic Programming and Evolvable Machines*, pp. 1–37, 2017.

[28] H. Beigy and M. R. Meybodi, “A learning automata-based adaptive uniform fractional guard channel algorithm,” *The Journal of Supercomputing*, vol. 71, no. 3, pp. 871–893, 2015.

[29] P. Venkata Krishna, S. Misra, D. Nagaraju, V. Saritha, and M. S Obaidat, “Learning automata based decision making algorithm for task offloading in mobile cloud,” in *International Conference on Computer, Information and Telecommunication Systems (CITS)*, Kunming, China, 2016, pp. 1–6.

[30] B. Damerchilu, M. S. Norouzzadeh, and M. R. Meybodi, “Motion estimation using learning automata,” *Machine Vision and Applications*, vol. 27, no. 7, pp. 1047–1061, 2016.

[31] L. Jiao, X. Zhang, B. J. Oomen, and O. Granmo, “Optimizing channel selection for cognitive radio networks using a distributed Bayesian learning automata-based approach,” *Applied Intelligence*, vol. 44, no. 2, pp. 307–321, 2016.

[32] D. E.Knuth, *The Art of Computer Programming*, vol. 2. Addison-Wesley, 1977.