On expediency of Closed Asynchronous Dynamic Cellular Learning Automata

Ali Mohammad Saghiri*, Mohammad Reza Meybodi

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

ARTICLE INFO

Article history:
Received 27 February 2017
Received in revised form 13 August 2017
Accepted 23 August 2017
Available online 30 August 2017

Keywords:
Cellular Automata
Learning Automata
Closed Asynchronous Dynamic Cellular Automata
Expediency

ABSTRACT

Closed Asynchronous Dynamic Cellular Learning Automata (CADCLAs) have been reported recently. CADCLAs are hybrid models based on Cellular Automata (CAs) and Learning Automata (LAs). Because of distributed computation characteristic of CAs and probabilistic decision making nature of LAs, analyzing the performance of CADCLAs based algorithms is difficult. The expediency metric has been used to study the performance of the LA based models. With respect to this metric, the performance of CADCLAs have not been studied in the literature. In this paper, we suggest sufficient conditions under them a CADCLA is expedient.

1. 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 neighborhoods 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. In a cell, the action selected by the LA of the cell is used to determine the state of that cell. Like CA, there is a local rule that the CLA operates under. For the LA of a cell, the local rule takes the actions selected by the neighboring LAs of that LA and then computes the reinforcement signal for that LA. The neighboring LAs (cells) of any particular LA (cell) constitute the local environment of that LA (cell). 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 [4–9]. In a DCLA, one of its aspects such as structure, local rule, or neighborhood may change over time. DCLAs can also be classified as Synchronous DCLAs or Asynchronous DCLAs. In synchronous DCLAs, all LAs in different cells are activated synchronously whereas in asynchronous DCLAs the LAs in different cells are activated asynchronously. All the reported DCLAs are asynchronous [10–12]. 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. CLAs have found applications in areas such as computer networks [11,13–15], social networks [9], Petri nets [16], and evolutionary computing [17], to mention a few.

The expediency metric has been widely used to study the performance 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

* Corresponding author.
E-mail addresses: a.m.saghiri@aut.ac.ir (A.M. Saghiri), mmeybodi@aut.ac.ir (M.R. Meybodi).

https://doi.org/10.1016/j.jocs.2017.08.015
1877-7503/© 2017 Elsevier B.V. All rights reserved.
each cell is replaced with a pure-chance automaton. Because of distributed computation characteristic and dynamicity of DCLAs, 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, we suggest sufficient conditions under them a CADCLA is expedient. The rest of this 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.

2. Learning Automata

In this section, the learning process of the LA is described. The LA 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 using an updating algorithm 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 quadruple <A, B, L, P>, where B is a set of reinforcement signals, A = {a1, a2, ..., an} is a set of actions, P denotes the action probability vector of the LA, and L is learning algorithm. The learning algorithm is used to update the probability vector of the LA.

\[
\begin{align*}
p_i(k+1) &= p_i(k) + a(1 - p_i(k)) \\
p_i(k+1) &= p_i(k) - ap_i(k), \quad \forall j \neq i \quad (1) \\
p_i(k+1) &= (1 - b)p_i(k) \\
p_i(k+1) &= \frac{b}{r - 1} + (1 - b)p_i(k), \quad \forall j \neq i \quad (2)
\end{align*}
\]

Let \(a_i\) be the action randomly chosen based on \(P(k)\) at step \(k\). In a linear learning algorithm, equation for updating probability vector \(P(k)\) is defined by \(1\) for a favorable response \((\beta = 1)\), and \(2\) for an unfavorable response \((\beta = 0)\). The probability of unfavorable response by action \(a_i\) is denoted by \(c_i = Pr[\beta = 0|a_i]\). 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 (decrease) 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 \((LP)\); if \(a > b\) the learning algorithm is called linear reward-penalty \((LARP)\); and finally if \(b = 0, \) it is called linear reward inaction \((LAI)\) algorithm. Learning automata have found applications in many areas such as sensor networks [10,11], stochastic graphs [20,21], peer-to-peer networks [22–27], channel assignment [28], mobile cloud computing [29], motion estimation [30], petri-nets [33], and cognitive networks [31] to mention a few.

3. 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.

3.1. 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 7-tuple \(CADCLA = (G, A, N, \Phi, \Psi, F_1, F_2)\), where:

- \(G = (V, E)\) is an undirected graph which determines the structure of CADCLA
- \(V = \{cell_1, cell_2, \ldots, cell_n\}\) is the set of vertices and \(E\) is the set of edges.
- \(A = \{LA_1, LA_2, \ldots, LA_n\}\) is a set of LAs each of which is assigned to one cell of CADCLA.
- \(N = \{N_1, N_2, \ldots, N_n\}\) where \(N_i = \{cell_j \in V | dist(cell_i, cell_j) < \theta_i\}\), where \(\theta_i\) 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 \(G\).
- \(\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_n\}\) where \(\Phi_i = (j, x_j)|cell_j \in N_i\) denotes the attribute of \(cell_j\), where \(x_j \subseteq \{x_1, x_2, \ldots, x_s\}\) is the set of allowable attributes. \(\Phi_i\) determines the attribute of \(cell_i\) when \(\theta_i = 1\).
- \(\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_n\}\) where \(\Phi_i = (j, alpha)|cell_j \in N_i\) and action \(alpha_i\) has been chosen by \(LA_j\) denotes the state of \(cell_j\). \(\Phi_i\) determines the state of \(cell_j\) when \(\theta_i = 1\).
- \(F_1 : (\Phi, \Psi) \rightarrow (\beta, \xi)\) is the local rule. In each cell, the local rule computes the reinforcement signal and restructuring signal for the cell based on the states and attributes of that cell and its neighboring cells. For example, in \(cell_i\), local rule takes \(\Phi_i, \Psi_i\) and returns \(\beta, \xi\). The LA of \(cell_i\) uses the reinforcement signal \(\beta_i\) to update its probability vector and the structure updating rule use the restructuring signals\(\alpha_i\) to change the neighborhood of \(cell_i\), the structure updating rule is described as below.
- \(F_2 : (\Phi, \Psi, \xi) \rightarrow (N_1)\) 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 \(N_1, \Psi_i, \xi_i\) and returns \(N_1\).

The internal structure of \(cell_i\), and its interaction with local environments is shown in Fig. 1.

3.2. The updating process of CADCLAs

The updating process of CADCLAs is described as follows. 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 as below.

1. Preparation phase: In this phase, a cell performs the following steps.
   Step 1: The cell sets its attribute.
   Step 2: The cell computes its restructuring signal using the local rule \((F_1)\).

2. Structure updating phase: In this phase, a cell performs the following step.
Step 1: The neighborhood structure of the cell is updated using the structure updating rule (F2) if the value of the restructuring signal of that cell is equal to 1.

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

Step 1: The LA of the cell selects one of its actions. The action selected by the LA in the cell determines the new state for that cell.

Step 2: The local rule (F1) is applied and a reinforcement signal is generated to update the probability vector of the LA of the cell.

During the update process, the structure updating rule and local rule are designated to find a cellular structure which receives low values for the restructuring signals in the cells. In addition, the LAs and local rule are designated to find a set of states for the cells which receives high values for the reinforcement signals for the LAs in the cells.

3.3. The performance metrics of CADCLAs

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

3.3.1. Entropy

The entropy of the CLA at iteration t is defined by Eq. (3) given below

\[ H(t) = -\sum_{k=1}^{n} r_k \times \ln(p_{kl}(t)) \]  

(3)

In the Eq. (3), n is the number of LAs of the CADCLA. \( r_k \) is the number of actions of the LAs. \( p_{kl}(t) \) is the probability of selecting action \( \alpha_l \) of the LA \( \alpha_l \) 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 [10, 11].

3.3.2. Potential energy

The potential energy of the CLA is defined by Eq. (4) given below

\[ T(t) = \sum_{i=1}^{n} \zeta(t) \]  

(4)

where \( \zeta(t) \) 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 T(t) indicates higher disorder in the structure of CLA.

4. 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 \( S(t) = (N(t), P(t), \Phi(t)) \) where

- \( N(t) = (N_1(t), N_2(t), \ldots, N_i(t)) \)
- \( P(t) = (P_1(t), P_2(t), \ldots, P_n(t)) \) where \( P_i(t) = (p_{i1}(t), p_{i2}(t), \ldots, p_{in}(t)) \) in which \( p_{ij}(t) \) is the probability of selecting action \( \alpha_j \) of the learning automaton \( LA_i \) at iteration \( t \). Each learning automaton has \( r \) actions.
- \( \Phi(t) = (\Phi_1(t), \Phi_2(t), \ldots, \Phi_n(t)) \)

The initial configuration of the CADCLA denoted by \( S(0) = (N(0), P(0), \Phi(0)) \). 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 \( \{S(t)\}_{t=0}^{\infty} \) which \( (S(t + 1) = \Phi(S(t), P(t), \Xi(t)) \).

**Definition 2.** A CADCLA is said to be expedient with respect to the cell \( cell_i \), if the following inequality holds:

\[ \lim_{t \to \infty} E \left[ D_{\Phi}^i(S(t)) \right] > \lim_{t \to \infty} D_{PE}^i(S_{PE}(t)) \]  

(5)

Before we define the elements of the above inequality, 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 \( \Phi^i_s, \Psi^i_s \) where \( \Phi^i_s \) is a version of \( \Phi^i \) which its item \( (i,-) \) is replaced with \( (i, \alpha_j) \) and \( \Psi^i_s \) is a version of \( \Psi^i \) which its item \( (i,-) \) is replaced with \( (i, b_k) \). Note that \( (i,-) \) refers to every item which its first element is equal to i.
- \( q_{ijk}^i(S(t)) \) denotes the reward probability of action \( \alpha_j \) of learning automaton \( LA_i \) when the index of attribute of cell \( i \) is \( k \). \( q_{ijk}^i(S(t)) \) is defined by Eq. (6) as given below

\[ q_{ijk}^i(S(t)) = E \left[ \beta_i = 1 \mid (i, \alpha_j) \in \Phi^i(t), (i, b_k) \in \Psi^i(t) \right] = \sum_j \sum_k \left( UG(i, j, k, t) \times F^i_k(UG(i, j, k, t)) \right) \]  

(6)

- \( d_{ijk}^i(S(t)) \) denotes the reward probability of action \( \alpha_j \) of learning automaton \( LA_i \). \( d_{ijk}^i(S(t)) \) is defined by Eq. (7) as given below

\[ d_{ijk}^i(S(t)) = \sum_k q_{ijk}^i(S(t)) \]  

(7)

Now we define the \( D_{\Phi}^i(S(t)) \) and \( D_{PE}^i(S_{PE}(t)) \)

\[ D_{\Phi}^i(S(t)) = E \left[ \beta_i = 1 \mid (\Phi^i(t), \Psi^i(t)) \right] = \sum_j \left( p_{ij}(t) \times d_{ijk}^i(S(t)) \right) \]  

(8)

\[ D_{PE}^i(S_{PE}(t)) \] where \( S_{PE}(t) = (P_{PE}(t), N_{PE}(t), \Phi^i_{PE}(t)) \) denotes the average reward received by cell \( i \) of 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.
The probability vectors of pure-chance automata of the CADCLA remains unchanged during the iterations of the CADCLA. Note that, \(L_P(i, j, k, t)^{\infty}\) is the probability of appearing the set \(L_P(i, j, k, t)^{\infty}\) 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 \(e\)-optimal with respect to cells, if for each cell, in which \(d_{ik}^l(S) > d_{ik}^l(S)\) where \(l \neq k\), we have the following inequality.

\[
\lim_{t \to \infty} \inf p_{ij}(t) > 1 - e \quad \text{w.p.1}
\]

**Definition 5.** The structure updating rule is called potential-decreasing, if applying this rule in each cell leading to decreasing the value of restructuring signal of that cell.

**Proposition 1.** if the structure updating rule is potential-decreasing and each cell is activated infinite times, then there is an iteration \(t' < t\) which \(T(t) = 0\).

**Proof.** In each iteration, utilizing potential-decreasing structure updating rule leading to approaching the value of restructuring signal of a cell to zero. Since each cell is activated infinite times, the value of restructuring signal of that cell ultimately changes to zero. Note that, the potential energy is calculated by summation of restructuring signals of all cells. Therefore, there is an iteration \(t'\) in which the potential energy is decreased to zero. In other word, the cellular structure approaches to a fixed structure because no cell changes its neighbors after iteration \(t'\). Note that a cell changes its neighbors if the value of its restructuring signal is equal to 1.

**5. Expediency of CADCLAs**

The expediency of the CADCLA has been studied in this section. In this section, the environment under which \(L_{LA}\) in the CLA is operating can be modeled as follows.

1. There is function \(f_{ij}^P(P_i)\) which \(f_{ij}^P(P_i(t)) = d_{ij}^l(S(t))\) and it is continuous in \(p_{ik}\) (\(j = 1, 2, \ldots, r\)).
2. \(\frac{d f_{ij}^P(P_i)}{dp_{ij}} < 0\)
3. \(\frac{d f_{ij}^P(P_i)}{dp_{ij}} \neq \frac{d f_{kj}^P(P_i)}{dp_{kj}}\) for \(k \neq j\)
4. \(f_{ij}^P(.)\) is continuously differentiable in all its arguments.
5. \(f_{ij}^P(P_i)\) and \(\frac{d f_{ij}^P(P_i)}{dp_{ij}}\) are Lipschitzian functions of all their arguments.

In this section, during evolution of the CADCLA, we assume that each cell is activated infinite times and \(a \to 0\) where \(a\) is the reward parameter.

**5.1. Expediency of CADCLA with \(L_{GP}\) Learning Algorithm for the LAs**

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

**Lemma 1.** If the non-stationary environment of a LA with \(L_{GP}\) Learning Algorithm satisfies the following conditions

- \(p_{ij}(1, 2, \ldots, \tau)\) is continuous in \(p_{ij}\) (\(j = 1, 2, \ldots, \tau\)).
- \(p_{ij}(1, 2, \ldots, \tau)\) denotes the penalty probability of choosing action \(\alpha_i\) and \(p_{ij}\) denotes the probability of choosing action \(\alpha_i\).
- \(\frac{d c_i(.)}{dp_{ij}} > 0\)
- \(\frac{d c_i(.)}{dp_{ij}} < \frac{d c_i(.)}{dp_{ij}}\) for \(j \neq i\)

- \(c_i(.)\) is continuously differentiable in all its arguments.

and \(a \to 0\) where \(a\) is the reward parameter, Then, the process \(\{p_{ij}(t)\}_{t=0}^\infty\) is Markovian and ergodic, which satisfy the following equation.

\[
\lim_{t \to \infty} E[p_{ij}(t) - p_{ij}^*]^2 = 0
\]

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

\[
p_{ij}^* \times c_i(p^*) = p_{ij}^* \times c_i(p^*) = \ldots = p_{ij}^* \times c_i(p^*)
\]

**Proof.** The proof is given in Section (7) of [19].

**Theorem 1.** In a CADCLA, if the learning algorithms of LAs are \(L_{GP}\), then regardless to the initial configuration, local rule, and structure updating rule, we have the following equation.

\[
\lim_{t \to \infty} (P(t)) = P^*
\]

**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, tries to find probability vector \(P_i^* = (p_{i1}^*, p_{i2}^*, \ldots, p_{ir}^*)\) which satisfy the Eq. (13). This phenomenon occurs in all cells and therefore the probability vectors of all LAs approach to \(P^*\) and the proof is completed.

\[
p_{i1}^* \times (1 - f_{i11}(P_i^*)) = p_{i2}^* \times (1 - f_{i21}(P_i^*)) = \ldots = p_{ir}^* \times (1 - f_{ir1}(P_i^*))
\]

**Lemma 2.** In a CADCLA, if the learning algorithms of LAs are \(L_{GP}\), then regardless to the initial configuration, local rule, and structure updating rule, we have \(1 - (r \times w_i) = \sum_{j=1}^{r} p_{ij}^* \times f_{ij}^P(P_i^*)\) where \(w_i = p_{i1}^* \times (1 - f_{i11}(P_i^*))\) which \(j \in \{1, \ldots, r\}\).

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

\[
p_{i1}^* \times (1 - f_{i11}(P_i^*)) = p_{i2}^* \times (1 - f_{i21}(P_i^*)) = \ldots = p_{ir}^* \times (1 - f_{ir1}(P_i^*)) = w_i
\]

**Eq. (15) is obtained by using Eq. (14).**

\[
(p_{i1}^* + p_{i2}^* + \ldots + p_{ir}^*) - (p_{i1}^* \times f_{i11}(P_i^*) + p_{i2}^* \times f_{i21}(P_i^*) + \ldots + p_{ir}^* \times f_{ir1}(P_i^*)) = r \times w_i
\]

**Theorem 2.** In a CADCLA, if the learning algorithms of LAs are \(L_{GP}\), and for every cell, \(w_i < \frac{1}{r} \sum_{j=1}^{r} f_{ij}^P(P_i^*)\), then the CADCLA is expedi-
dient regardless to its initial configuration, local rule, and structure updating rule.

**Proof.** In order to prove that a CADCLA is expedient with respect to cells, we need to show that the inequality \( \lim_{t \to \infty} E \left[ D^t_1(S(t)) \right] > \lim_{t \to \infty} E \left[ D^t_y(S(t)) \right] \) holds for every cell \( i \). By expanding both sides of this inequality, we have the following inequality.

\[
\lim_{t \to \infty} E \left( \sum_{j=1}^{r} \left( p_{ij}(t) \times f^\beta_{ij}(P_i(t)) \right) \left( p_{ij}(t) \times f^\alpha_{ij}(P_j(t)) \right) \right) > \lim_{t \to \infty} E \left( \sum_{j=1}^{r} \left( p_{ij}(t) \times f^\beta_{ij}(P_j(t)) \right) \left( p_{ij}(t) \times f^\alpha_{ij}(P_j(t)) \right) \right)
\]

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 \( i \) of \( D^t_1(t) \) is omitted in Eq. (20).

\[
\lim_{t \to \infty} E \left[ p_{11}(t) \times f^\beta_{11}(P_1(t)) + p_{12}(t) \times f^\beta_{12}(P_1(t)) \right. \\
\left. + \ldots + p_{1r}(t) \times f^\beta_{1r}(P_1(t)) \right] > \\
\lim_{t \to \infty} E \left[ p_{11}(t) \times f^\beta_{11}(P_1(t)) + p_{12}(t) \times f^\beta_{12}(P_1(t)) \right. \\
\left. + \ldots + p_{1r}(t) \times f^\beta_{1r}(P_1(t)) \right]
\]

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

\[
\lim_{t \to \infty} E \left[ p_{11}(t) \times f^\beta_{11}(P_1(t)) + p_{12}(t) \times f^\beta_{12}(P_1(t)) \right. \\
\left. + \ldots + p_{1r}(t) \times f^\beta_{1r}(P_1(t)) \right] > \\
\lim_{t \to \infty} E \left[ p_{11}(t) \times f^\beta_{11}(P_1(t)) + p_{12}(t) \times f^\beta_{12}(P_1(t)) \right. \\
\left. + \ldots + p_{1r}(t) \times f^\beta_{1r}(P_1(t)) \right]
\]

After replacing \( p^\beta_{1i}(t) \) with \( \frac{1}{r} \) in the right hand side, Eq. (21) changes to Eq. (22).

\[
\lim_{t \to \infty} E \left[ p_{11}(t) \times f^\beta_{11}(P_1(t)) + p_{12}(t) \times f^\beta_{12}(P_1(t)) \right. \\
\left. + \ldots + p_{1r}(t) \times f^\beta_{1r}(P_1(t)) \right] > \\
\sum_{i=1}^{r} \frac{f^\beta_{1i}(P_i)}{r}
\]

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

\[
\sum_{j=1}^{r} \left( p_{ij}^\beta \times f^\beta_{1i}(P_i) \right) > \frac{\sum_{j=1}^{r} f^\beta_{1i}(P_i)}{r}
\]

By substituting Eq. (18) in Eq. (23) we have the following.

\[
1 - (r \times w_i) > \frac{\sum_{j=1}^{r} f^\beta_{1i}(P_i)}{r}
\]

\[
w_i < \frac{1}{r} - \frac{\sum_{j=1}^{r} f^\beta_{1i}(P_i)}{r}
\]

And the proof is complete.

In other word, not only the conditions mentioned in this section for the environment of the cells must be checked but also the condition on \( w_i \) which introduced by Theorem 2 must be checked to determine whether the CADCLA can be expedient or not. Note that, the value of the right hand side of inequality (25) can be computed because it is not depending on time. The computed value determines an upper bound for \( w_i \). As it was previously mentioned, \( w_i = p_{ii}^\beta \times (1 - f^\beta_{1i}(P_i)) \) which \( j \in \{1, \ldots, r\} \).

**Proposition 2.** If the conditions mentioned in Theorem 2 are satisfied, then by proper choice of parameters of the LASs, 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 \( \lim_{t \to \infty} (H(t)) \) we have

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

According to the result of Theorem 1, we have \( \lim_{t \to \infty} (P(t)) = P^* \). Therefore, \( H(t) \) approaches to \( - \sum_{k=1}^{n} \sum_{l=1}^{r_k} \left[ p_{kl}^* \times \ln(p_{kl}^*) \right] \) 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 \( \lim_{t \to \infty} (T(t)) = 0 \) and \( \lim_{t \to \infty} (H(t)) = h^* \) where \( h^* \) is a constant value.

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

5.2. Expedient of CADCLA with \( L_{R,P} \) Learning Algorithm for the LASs

In this section, we suggest a set of conditions under which the CADCLA with \( L_{R,P} \) learning algorithm for the LASs is expedient. In this section, an important assumption is considered as described as follow. In every cell, we assume that there is an action \( \alpha_i \) which \( f_{ij}^\beta(P_i) > f_{ij}^\beta(P_j) \) where \( i \neq k \). In other word, action \( \alpha_i \) has the highest reward probability for every probability vector of the LA in cell.4.

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

- \( c_i(p_1, p_2, \ldots, p_r) \) is continuous in \( p_j(i,j = 1,2,\ldots,r) \).
- \( \frac{\delta c_i}{\delta p_j} > 0 \)
- \( \frac{\delta c_i}{\delta p_j} \leq \frac{\delta c_j}{\delta p_j} \) for \( j \neq i \)
- \( c_i(.) \) is continuously differentiable in all its arguments.
- \( c_i(.) \) and \( \left( \frac{\delta c_j}{\delta p_j} \right) \) are Lipschitz functions of all their arguments.
- There exist an action \( \alpha_i \) which \( c_i(p) > c_i(p) \) for \( k \neq l \).

Then, by proper choice of parameters in the LASs and for any given \( \varepsilon > 0 \), the process \( \{ p(t) \} \) is Markovian and ergodic, which satisfy the following inequality.

\[
\lim_{t \to \infty} \inf p_{ij}(t) > 1 - \varepsilon \quad \text{w.p.1}
\]

Where \( p(t) \) denotes the probability of selecting action \( \alpha_i \) 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 \( \varepsilon > 0 \) and by proper choice of parameters in the LASs, if the learning algorithms of LASs are \( L_{R,P} \), then regardless to its initial configuration, local rule, and structure updating rule, the CADCLA is \( \varepsilon \)-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 \( \alpha_i \) of \( LA_i \) \( \{cell\} \) which \( \alpha_i \) 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 \( \varepsilon \)-optimal with respect to cells according to definition 4 and the proof is complete.

**Theorem 4.** If the learning algorithms of LASs are \( L_{R,P} \), and for every cell such as \( cell \), we have

\[
\lim_{t \to \infty} E \left[ p_{ij}(t) \times f_{ij}^\beta(P_i(t)) \right] > \frac{\sum_{j=1}^{r} f_{ij}^\beta(P_i)}{r^2}.
\]

Then, the CADCLA is expedient with...
respect to cells regardless to its initial configuration, local rule, and structure updating rule.

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

\[
\lim_{t \to \infty} E[D^\beta (S(t))] > \lim_{t \to \infty} \sum_{i=1}^n (p^\beta (t) \times f^\beta (P^\beta (t)))
\]

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 \( p^\beta (t) \) in right hand side of (27) with \( \frac{1}{t} \), we have the following inequality.

\[
\lim_{t \to \infty} E[p^\beta (t) \times f^\beta (P(t))] > \lim_{t \to \infty} \sum_{j=1}^n p^\beta (t) - \lim_{t \to \infty} \sum_{j \neq 1} p^\beta (t) \times f^\beta (P(t))
\]

And the proof is complete.

In other word, not only the conditions mentioned in this section for the environment of the cells must be checked but also the condition over the probability vectors of the LAs which introduced by Theorem 4 must be checked to determine whether the CADCLA can be expedient or not. Note that, the value of \( \sum_{i=1}^n p^\beta (t) \) of inequality (28) can be computed because it is not depending on time.

**Proposition 4.** If the conditions mentioned in Theorem 4 are satisfied and the parameters of the LAs are chosen properly, then \( \lim_{t \to \infty} H(t) = h^\beta \) where \( h^\beta \) is a value which depends on \( \varepsilon \).

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

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

\[
\lim_{t \to \infty} \left( \sum_{k=1}^n [p_{k1}(t) \times \ln(p_{k1}(t))] + p_{k2}(t) \times \ln(p_{k2}(t)) + \ldots + p_{kn}(t) \times \ln(p_{kn}(t)) \right)
\]

As a result of Theorem 3, by proper choice of parameters in the LAs, we have \( \lim_{t \to \infty} p_{kl}(t) > 1 - \varepsilon \) with probability 1 for action \( \alpha_1 \) of LA. Note that, if the value of \( p_{kl}(t) \) approaches to a value higher than \( 1 - \varepsilon \), the summation of probabilities of selecting other actions of LA approaches to a value lower than \( \varepsilon \). This phenomenon occurs in all LAs of the CADCLA and therefore, \( \lim_{t \to \infty} H(t) \) approaches to \( h^\beta \) which \( h^\beta \) is a value which depends on \( \varepsilon \). It is obvious that if \( \varepsilon \) is very close to zero, the value of \( h^\beta \) is very close to zero.

**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 \( \lim_{t \to \infty} H(t) = h^\beta \) and \( \lim_{t \to \infty} T(t) = 0 \).

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

---

6. Simulation

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

6.1. Simulation environment and setup

Fig. 2 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 \( \{(1, "RED"),(2, "BLUE"),(3, "RED"),(4, "BLUE"),(5, "RED") \} \) respectively. The values of \( \Phi^1_1 (0), \Phi^2_1 (0), \Phi^3_1 (0), \Phi^4_1 (0) \) are set to \( \{(1, "ON"),(2, "ON"),(3, "ON"),(4, "ON"),(5, "ON") \} \) 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. An 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.

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 have similar attributes with that cell. If the value returned by the similarity function is lower than a threshold 0.5, the restructuring signal is set to 1 and 0 otherwise. At the end of preparation phase, the cell goes to the 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 neighbor 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 LA of the cell selects one of its actions. The action selected by the LA 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 \( D(t) = [d_{ij}(t)]_{5 \times 2} \) denotes the reward probabilities matrix in which \( d_{ij}(t) \) denoted the reward probability of learning automaton LA and action \( \alpha_j \) in iteration \( t \).

\[
D(t) = \begin{bmatrix}
(1 - p_{11}(t)) & p_{11}(t) \\
p_{21}(t) & (1 - p_{21}(t)) \\
p_{32}(t) & (1 - p_{32}(t)) \\
p_{41}(t) & (1 - p_{41}(t)) \\
p_{51}(t) & (1 - p_{51}(t))
\end{bmatrix}
\]
6.2. Experiment 1

This experiment is conducted to confirm that if the condition mentioned in proposition 5 are satisfied, then we have \( \lim_{t \to \infty} (T(t)) = 0 \) and \( \lim_{t \to \infty} (H(t)) = h^* \). In this experiment, the learning algorithms of LAs are \( L_{R_P} \). The reward parameter and penalty parameter of the LAs are set to 0.001 and 0.00001 respectively. Some conditions mentioned for proposition 5 are satisfied in this experiment. The results given in Fig. 3 and Fig. 4 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.

6.3. Experiment 2

This experiment is conducted to confirm that if the condition mentioned in proposition 3 are satisfied, then we have \( \lim_{t \to \infty} (T(t)) = 0 \) and \( \lim_{t \to \infty} (H(t)) = h^* \) 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. Some conditions mentioned for proposition 3 are satisfied in this experiment. The results given in Fig. 5 and Fig. 6 show that the values of Potential Energy of CADCLA approach to zero. Therefore, the cellular structure approaches to a fixed structure.

7. Conclusion

In this paper, we suggested sufficient conditions under them a CADCLA is expedient. This study is applicable for both \( L_{RP} \) and \( L_{RP} \) 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 \( L_{RP} \) learning algorithm for the LAs. In addition, we plan to define new metrics to evaluate the behavior of the CADCLA.

References


Ali Mohammad Saghiri received the B. Sc. and M. Sc. degrees in computer engineering in Iran, in 2008 and 2010, respectively. He is currently the Ph.D. student of computer engineering in AmirKabir University of Technology, Tehran, Iran. His research interests include peer-to-peer networks, distributed systems, artificial intelligence, learning automata, reinforcement learning, parallel algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, AmirKabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft-computing and software development.