9th Annual Computer Society of Iran Computer Conference
Sharif University of Technology
February 17-19, 2004

A Mathematical Framework for Cellular Learning Automata

Hamid Beigy and M. R. Meybodi

Soft Computing Laboratory,
Computer Engineering Department,
Amirkabir University of Technology,
Tehran, IRAN

Abstract

In this paper, we first give a formal description for cellular learning automata and then study its convergence behavior. It is shown that for permutable rules, the cellular learning automata converges to a stable and compatible configuration. The numerical results also confirms our theoretical investigations.

1 Introduction [1]

Decentralization is a common feature of natural and man-made systems in which, due to large spatial separation of decision makers or limited bandwidth of communication channels, complete information exchange may not be feasible. The decision makers in such a system can gather limited information about each other and the overall system. Hence, the decisions must be made by individual decision makers that have access to partial information regarding the state of the system. Decentralization, by its nature, introduces uncertainty in to the decision process.

In addition to spatial separation of system and incomplete information exchange, uncertainties regarding system parameters, control actions taken by other decision makers and external events increase the complexity of decentralized systems. Even in the absence of these uncertainties it is well known that the coordination of decentralized decision makers is a formidable problem.

Adaptation (learning) in decision process overcomes the introduced uncertainty. By using learning, the different decentralized decision makers used in the system attempt to converge to their optimal strategies by improving their performance online, based upon the response of the overall system. Hence, learning can be considered as a critical part of decision makers that have access to the partial information. A subclass of such systems, which are modelled using cellular automata (CA), use information exchange with neighborhood decision makers.

Cellular automata are mathematical models for systems consisting of large numbers of simple identical components with local interactions. CA are non-linear dynamical systems in which space and time are discrete. It is called *cellular*, because it is made up cells like points in the lattice or like squares of the checker boards and it is called *automata*, because it follows a simple rule [1]. The simple components act together to produce complicated patterns of behavior. Cellular automata perform complex computation with high degree of efficiency and robustness. They are specially suitable for modelings natural systems that can be described as massive collections of simple objects interacting locally with each other [2][3]. Informally, a $d-$dimensional CA consists of an infinite $d-$dimensional lattice of identical cells. Each cell can assume a state from finite set of states. The cells update their states synchronously on discrete steps according to a local rule. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighborhood [4]. The state of all cells in the lattice are described by a configuration. A configuration can be described as the state of the whole lattice. The rule and the initial configuration of the CA specifies the evolution of CA that tells how each configuration is changed in one step. Formally, a CA can be defined as follows

Definition 1 *A $d-$dimensional cellular automata is a structure $A = (Z^d, \Phi, N, F)$, where*

(1) Z^d is a lattice of $d-$tuples of integer numbers.

[1] This research was in part supported by grant from IPM (No. CS1382-5-04).

Each cell in the d-dimensional lattice, Z^d, is represented by a d-tuple $(z_1, z_2, \cdots, z_d)$.

(2) $\Phi = \{1, \cdots, m\}$ *is a finite set of states.*

(3) $N = \{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_m\}$ *is a finite subset of* Z^d *called neighborhood vector, where* $\bar{x}_i \in Z^d$. *The neighborhood vector determines the relative position of the neighboring lattice cells from any given cell u in the lattice* Z^d. *The neighbors of a particular cell u are set of cells* $\{u + \bar{x}_i | i = 1, 2, \cdots, \bar{m}\}$. *We assume that, there exists a neighborhood function* $\bar{N}(u)$ *that maps a cell u to the set of its neighbors, that is*

$$\bar{N}(u) = (u + \bar{x}_1, u + \bar{x}_2, \cdots, u + \bar{x}_{\bar{m}}). \quad (1)$$

For the sake of simplicity, we assume that the first element of neighborhood vector (i.e. $\bar{x}_1$*) is equal to d-tuple* $(0, 0, \cdots, 0)$ *or equivalently* $u + \bar{x}_1 = u$. *The neighborhood function* $\bar{N}(u)$ *must satisfy in the two following conditions:*

- $u \in \bar{N}(u)$ *for all* $u \in Z^d$.
- $u_1 \in \bar{N}(u_2) \iff u_2 \in \bar{N}(u_1)$ *for all* $u_1, u_2 \in Z^d$.

(4) $F : \Phi^{\bar{m}} \to \Phi$ *is the local rule of the cellular automata. It gives the new state of each cell from the current states of its neighbors.*

Learning in the learning automata have been studied using the paradigm of an automaton operating in an unknown random environment. In a simple form, the automaton has finite set of actions to choose from and at each stage, its choice (action) depends upon its action probability vector. For each action chosen by the automaton, the environment gives a reinforcement signal with fixed unknown probability distribution. The automaton then updates its action probability vector depending upon the reinforcement signal at that stage, and evolves to the some final desired behavior. Learning automata can be classified into two main families: *fixed structure learning automata* and *variable structure learning automata* [5]. Variable structure learning automata are represented by triple $< \beta, \alpha, T >$, where β is a set of inputs, α is a set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation and is used to modify action probability vector $\underline{p}$. Various learning algorithms have been reported in the literature. In what follows, two learning algorithms for updating the action probability vector are given. Let α_i be the action chosen at time k as a sample realization from probability distribution $p(k)$. In linear reward-inaction algorithm (L_{R-I}) scheme the recurrence equation for updating p is defined as

$$p_j(k+1) = \begin{cases} p_j(k) + a \times [1 - p_j(k)] & \text{if} \quad i = j \\ p_j(k) - a \times p_j(k) & \text{if} \quad i \neq j \end{cases}$$

when $\beta(k) = 0$ and the action probability vector remains unchanged when $\beta(k) = 1$. Parameters $0 < a < 1$ represent *step lengths* and r is the number of actions for LA. LA have been used successfully in many applications such as telephone and data network routing [6], solving NP-Complete problems [7], capacity assignment [8] and neural network engineering [9,10] to mention a few.

Automata are, by design, "simple agents for doing simple things". The full potential of a LA is realized when multiple automata interact with each other. Interaction may assume different forms such as tree, mesh, array and etc. Depending on the problem that needs to be solved, one of these structures for interaction may be chosen. In most applications, full interaction between all LAs is not necessary and is not natural. Local interaction of LAs, which can be defined in a from of graph such as tree, mesh, or array, is natural in many application. In the other hand, CA are mathematical models for systems consisting of large numbers of simple identical components with local interactions. In this paper, we combine the CA and LA to obtain a new model called cellular learning automata (CLA). This model is superior to CA because of its ability to learn and also is superior to single LA because it is a collection of LAs which can interact with each other. The basic idea of CLA, which is a subclass of stochastic CA, is to use learning automata (LA) to adjust the state transition probability of stochastic CA. The CLA can be classified into *synchronous* and *asynchronous*. In synchronous CLA, all cells are synchronized with a global clock and executed at the same time. In [11], an asynchronous CLA with several LA in each cell is given and used as an adaptive controller. In this model, state space of system under control is uniformly discretized into cells. The actions of each LA correspond to discritized values of the corresponding control variable. Based on the state of system (S_0) one cell in CLA is activated. Every LA of the activated cell chooses an action based on its action probability vector. These actions are applied to the system and the state of system is changed from S_0 to S_1. The environment then passes a reinforcement signal to the LA of the activated cell. Depending on this signal, LA in activated cell and its neighboring cells update their action probability vectors. This process continues until termination state is reached. In [12], a model of synchronous CLA has been proposed in which each cell can hold one LA. The CLA have been used in many applications such as image processing [13], rumor diffusion [14], image processing [15–18], modelling of commerce networks [19], channel assignment in cellular networks [20], and VLSI Placement [21] to mention a few.

Since introduction of CLA, it has been used in a number of applications but no mathematical framework for studying its behavior has been developed yet. Having a mathematical framework for CLA not only enables us to investigate the characteristics of this model deeper, which may lead us to find more applications, but having such a mathematical framework also makes it possible to study the previous applications more rigor-

ously and develop better CLA based algorithms for those applications. In this paper, we develop a mathematical framework for studying the behavior of the CLA and investigate its convergence properties. It is shown that for commutative rules, the CLA converges to a globally stable state.

The rest of this paper is organized as follows. In section 2, the CLA is presented. The synchronous CLA and the synchronous CLA using commutative rules are presented in sections 3 and 4, respectively. Sections 5 presents the numerical example and section 6 concludes the paper.

2 Cellular Learning Automata

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

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

Definition 2 *A $d-$dimensional cellular learning automata is a structure $A = (Z^d, \Phi, A, N, F)$, where*

- *Z^d is a lattice of $d-$tuples of integer numbers.*
- *$\underline{\Phi}$ is a finite set of states.*
- *A is the set of LAs each of which is assigned to each cell of the CA.*
- *$N = \{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_m\}$ is a finite subset of Z^d called neighborhood vector, where $\bar{x}_i \in Z^d$.*
- *$F : \underline{\Phi}^m \rightarrow \underline{\beta}$ is the local rule of the cellular automata, where $\underline{\beta}$ is the set of values that the reinforcement signal can take. It gives the reward (reinforcement) signal to each LA from the current actions selected by its neighboring LAs.*

In what follows, we consider CLA with n cells and neighborhood function $\bar{N}(i)$. A learning automaton denoted by A_i, which has a finite action set $\underline{\alpha}_i$, is associated to cell i (for $i = 1, \ldots, n$) of CLA. Let cardinality of $\underline{\alpha}_i$ be m_i and the state of CLA represented by $\underline{p} = (\underline{p}'_1, \underline{p}'_2, \ldots, \underline{p}'_n)'$, where $\underline{p}_i = (p_{i1}, \ldots, p_{im_i})'$ is the action probability vector of A_i. It is evident that the local environment for each learning automaton is the learning automata residing in its neighboring cells. From the repeated application of simple local rules and simple learning algorithms, the global behavior of CLA can be very complex.

The operation of CLA takes place as the following iterations. At iteration k, each learning automaton chooses an action. Let $\alpha_i \in \underline{\alpha}_i$ be the action chosen by A_i. Then all learning automata receive a reinforcement signal. Let $\beta_i \in \underline{\beta}$ be the reinforcement signal received by A_i. This reinforcement signal is produced by the application of local rule $F^i(\alpha_{i+\bar{x}_1}, \alpha_{i+\bar{x}_2} \ldots, \alpha_{i+\bar{x}_m}) \rightarrow \underline{\beta}$. The higher value of β_i means that the chosen action of A_i is more rewarded. Since each set $\underline{\alpha}_i$ is finite, rule $F^i(\alpha_{i+\bar{x}_1}, \alpha_{i+\bar{x}_2} \ldots, \alpha_{i+\bar{x}_m}) \rightarrow \underline{\beta}$ can be represented by a hyper matrix of dimensions $m_1 \times m_2 \times \cdots \times m_m$. These n hyper matrices together constitutes what we call the rule of CLA. When all of these n hyper matrices are equal, the rule is uniform; otherwise the rule is nonuniform. For the sake of simplicity in our presentation, the rule $F^i(\alpha_{i+\bar{x}_1}, \alpha_{i+\bar{x}_2} \ldots, \alpha_{i+\bar{x}_m})$ is denoted by $F^i(\alpha_1, \alpha_2 \ldots, \alpha_m)$. Based on set $\underline{\beta}$, the CLA can be classified into three groups: P-model, Q-model, and S-model Cellular learning automata. When $\underline{\beta} = \{0, 1\}$, we refer to CLA as *P-model cellular learning automata*, when $\underline{\beta} = \{b_1, \ldots, b_l\}$, (for $l < \infty$), we refer to CLA as *Q-model cellular learning automata*, and when $\underline{\beta} = [b_1, b_2]$, we refer to CLA as *S-model cellular learning automata*. If learning automaton A_i uses learning algorithm L_i, we denote CLA by the $CLA(L_1, \ldots, L_n)$. If $L_i = L$ for all $i = 1, \cdots, n$, then we denote the CLA by the $CLA(L)$.

In order to analyze the behavior of the CLA, in the following subsections we give some definitions and notations.

2.1 Definitions and Notations

In this subsection, we give some definition and derive some primary results regarding CLA which will be used later for the analysis of CLA.

Definition 3 *A configuration of CLA is a map $K : Z^d \rightarrow \underline{p}$ that associates an action probability vector with every cell. We will denote the set of all configurations of A by $K(A)$ or simply K.*

The application of the local rule to every cell allows transforming a configuration to a new one.

Definition 4 *The global behavior of a CLA is a mapping $G : K \rightarrow K$ that describes the dynamics*

of CLA.

Definition 5 *The evolution of CLA from a given initial configuration* $\underline{p}(0) \in K$ *is a sequence of configurations* $\{\underline{p}(k)\}_{k\geq 0}$, *such that* $\underline{p}(k+1) = G(\underline{p}(k))$.

CLA may be described as a network of LA assigned to the nodes of a graph (usually a finite and regular lattice). Each node is connected to a set of nodes (its neighbors), and each LA updates its action probability vector at discrete time instants, using a learning algorithm and a rule which depends on its own action and that of its neighbors.

Definition 6 *A CLA is called synchronous if all the LA are activated at the same time and asynchronous if at a given time only some LAs are activated.*

Definition 7 *A CLA is called uniform if the neighborhood function and the local rule are the same for all cells of CLA.*

Definition 8 *A rule is called uniform if the rule for all cells are the same; otherwise it is call nonuniform.*

Definition 9 *A configuration* $\underline{p}$ *is called deterministic if the action probability vector of each learning automaton is a unit vector; otherwise it is called probabilistic. Hence, the set of all deterministic configurations,* K^*, *set of probabilistic configurations,* K, *in CLA are*

$$K^* = \{\underline{p} | \underline{p} = (\underline{p}'_1, \underline{p}'_2, \ldots, \underline{p}'_n)', \underline{p}_i = (p_{i1}, \ldots, p_{im_i})', \quad p_{iy} = 0 \text{ or } 1 \ \forall y, i, \sum_y p_{iy} = 1 \ \forall i\}$$

and

$$K = \{\underline{p} | \underline{p} = (\underline{p}'_1, \underline{p}'_2, \ldots, \underline{p}'_n)', \underline{p}_i = (p_{i1}, \ldots, p_{im_i})', \quad 0 \leq p_{iy} \leq 1 \ \forall y, i, \sum_y p_{iy} = 1 \ \forall i\},$$

respectively.

In the rest of this section, we give some definition and derive some results regarding CLA which will be used later for the analysis of CLA.

Definition 10 *The average reward for action* r *of automaton* A_i *in a CLA with configuration* $\underline{p} \in K$ *is defined as*

$$d_{ir}(\underline{p}) = \sum_{y_2} \cdots \sum_{y_m} F^i(r, y_2, \ldots, y_m) \prod_{\substack{l \in \hat{N}(i) \\ l \neq i}} p_{ly_l}, \quad (2)$$

and the average reward for learning automaton A_i *is equal to*

$$D_i(\underline{p}) = \sum_r d_{ir}(\underline{p}) p_{ir}. \quad (3)$$

The above definition implies that if learning automaton A_j is not a neighboring learning automaton for A_i, then $d_{ir}(\underline{p})$ does not depend on $\underline{p}_j$.

Definition 11 *A configuration* $\underline{p} \in K$ *is compatible providing*

$$\sum_r d_{ir}(\underline{p}) p_{ir} \geq \sum_r d_{ir}(\underline{p}) q_{ir} \quad (4)$$

for all configurations $\underline{q} \in K$ *and all cells* i. *The configuration* $\underline{p} \in K$ *is said to be* fully compatible, *if the above inequalities are strict.*

The compatibility of a configuration implies that no learning automaton in CLA have any reason to change its action.

Definition 12 *A configuration* $\underline{p} \in K$ *is admissible providing*

$$D_i(\underline{p}) \geq D_i(\underline{q}) \quad (5)$$

for all configurations $\underline{q} \in K$ *and all cells* i.

The compatibility is a local concept and can be calculated by looking only into the neighboring learning automata, but the admissibility is a global concept.

Corollary 1 *Admissibility implies compatibility but the converse is not true, i.e, every admissible configuration is compatible but every compatible configuration is not necessarily admissible.*

Proof. The proof is trivial from definitions 11 and 12.

Definition 13 *Total average reward for the CLA, which is the sum of the average reward for all the learning automata in the CLA, for configuration* $\underline{p} \in K$ *is defined as*

$$D(\underline{p}) = \sum_i D_i(\underline{p}). \quad (6)$$

Corollary 2 *A configuration* $\underline{p} \in K$ *is admissible if and only if* $D(\underline{p}) \geq D(\underline{q})$ *for all* $\underline{q} \in K$.

Proof. The proof is trivial by looking definitions 12 and 13.

Remark 1 *With the approach given in this paper, the probability of different configurations are updated according to a learning algorithm that can be considered as hill-climbing in probability space. At every stage, the change in probabilities are such that total average reward is improved monotonically in expected sense.*

Lemma 1 *CLA has at least one compatible configuration.*

Proof. The proof of this Lemma is given in [22].

Lemma 2 *Let* $\underline{p} \in K$ *be a compatible configuration. Then for each* i, *we have*

$$d_{ir}(\underline{p}) = D_i(\underline{p}),$$

for all r *such that* $p_{ir} > 0$.

Proof. The proof of this Lemma is given in [22].

Lemma 3 *Configuration $\underline{p} \in K$ is compatible if and only if*

$$d_{ir}(\underline{p}) \leq D_i(\underline{p}),$$

for all i and r.

Proof. The proof of this Lemma is given in [22].

This lemma provides a means of finding compatible configurations in the CLA.

Theorem 1 *A configuration $\underline{p} \in K$ is compatible if and only if $\sum_i \sum_y d_{iy}(\underline{p})\,[p_{iy} - q_{iy}] \geq 0$ holds for all $\underline{q} \in K$.*

Proof. The proof of this Lemma is given in [22].

This Theorem states that, when the action probability vector of all learning automata except the specific A_i are held fixed, then the configuration reached by the CLA at the point, where the average reward of A_i is maximum, is compatible.

Theorem 2 *A corner $\underline{p} = \left(\underline{e}_{t_1}, \underline{e}_{t_2}, \ldots, \underline{e}_{t_n}\right)'$ is compatible if and only if*

$$F^i(t_1, t_2, \ldots, t_m) \geq F^i(r, t_2, \ldots, t_m)$$

for all $r \neq t_i$.

Proof. The proof of this Lemma is given in [22].

3 Synchronous Cellular Learning Automata

In this section, we analyze the synchronous CLA in which all learning automata use the L_{R-I} learning algorithm. We denote this CLA by S-CLA(L_{R-I}). Using L_{R-I} learning algorithm, process $\{\underline{p}(k)\}_{k\geq 0}$ is Markovian and can be described by the following difference equation.

$$\underline{p}(k+1) = \underline{p}(k) + \underline{a}\,\underline{g}(\underline{p}(k), \underline{\beta}(k)), \tag{7}$$

where $\underline{\beta}(k)$ is composed of components $\beta_{iy}(k)$ (for $1 \leq i \leq n$ and $1 \leq y \leq m_i$), which are dependent on $\underline{p}$. $\underline{g}$ represents the learning algorithm, $\underline{a}$ is a $M \times M$ diagonal matrix with $a_{jj} = a_i$ for $\sum_{l=1}^{i-1} m_l < i \leq \sum_{l=1}^{i} m_l$, and a_i represents the learning parameter for learning automaton A_i. Now, define

$$\Delta\underline{p}(k) = \mathrm{E}\left[\underline{p}(k+1)|\underline{p}(k)\right] - \underline{p}(k). \tag{8}$$

Since $\{\underline{p}(k)\}_{k\geq 0}$ is Markovian and $\underline{\beta}(k)$ depends only on $\underline{p}(k)$ and not on k explicitly, then $\Delta\underline{p}(k)$ can be given by a function of $\underline{p}(k)$. Hence, we can write

$$\Delta\underline{p}(k) = \underline{a}\,\underline{f}\left(\underline{p}(k)\right). \tag{9}$$

Now using L_{R-I} algorithm, the components of $\Delta\underline{p}(k)$ can be obtained as follows.

$$\begin{aligned}\Delta p_{iy}(k) &= a_i p_{iy}(k) \sum_{r\neq y} p_{ir}(k)\left\{\mathrm{E}\left[\beta_{iy}(k)\right] - \mathrm{E}\left[\beta_{ir}(k)\right]\right\} \\ &= a_i p_{iy}(k)\left[d_{iy}(\underline{p}) - D_i(\underline{p})\right] \qquad (10)\\ &= a_i f_{iy}(\underline{p}). \qquad (11)\end{aligned}$$

For different values of $\underline{a}$, equation (7) generates different process and we shall use $\underline{p}^a(k)$ to denote this process whenever the value of $\underline{a}$ is to be specified explicitly. Define a sequence of continuous-time interpolation of (7), denoted by $\tilde{\underline{p}}^a(t)$ and called *interpolated process*, whose components are defined by

$$\tilde{\underline{p}}_i^{\,a}(t) = \underline{p}_i(k) \qquad t \in [ka_i, (k+1)a_i], \tag{12}$$

where a_i is the learning parameter of the L_{R-I} algorithm for learning automaton A_i. The objective is to study the limit of sequence $\{\tilde{\underline{p}}^a(t)\}_{t\geq 0}$ as $\max\{\underline{a}\} \to 0$, which will be a good approximation to the asymptotic behavior of (12). When learning parameter a_i is sufficiently small for all $i = 1, 2, \ldots, n$, then equation (9) can be written as the following ordinary differential equation (ODE).

$$\dot{\underline{p}} = \underline{f}(\underline{p}), \tag{13}$$

where $\dot{\underline{p}}$ is composed of the following components.

$$\frac{dp_{iy}}{dt} = p_{iy}\left[d_{iy}(\underline{p}) - D_i(\underline{p})\right] \tag{14}$$

We are interested in characterizing the long term behavior of $\underline{p}(k)$ and hence the asymptotic behavior of ODE (13). The analysis of process $\{\underline{p}(k)\}_{k\geq 0}$ is done in two stages. In the first stage, we solve ODE (13) and in the second stage, we characterize the solution of this ODE. The solution of ODE (13) approximates the asymptotic behavior of $\underline{p}(k)$ and the characteristics of this solution specify the long term behavior of $\underline{p}(k)$. The following theorem gives the asymptotic behavior of $\tilde{\underline{p}}^a$ as $\max\{\underline{a}\}$ is sufficiently small. We show that the sequence of interpolated process $\{\tilde{\underline{p}}^a(t)\}$ converges weakly to the solution of ODE (13) with initial configuration $\underline{p}(0)$. This implies that asymptotic behavior of $\underline{p}(k)$ can be obtained from the solution of ODE (13).

Theorem 3 *Sequence $\{\tilde{\underline{p}}^a(.)\}$ converges weakly to the solution of*

$$\frac{d\underline{X}}{dt} = \underline{f}(\underline{X}) \tag{15}$$

with initial condition $\underline{X}(0) = X_0$ as $a \to 0$, where $X_0 = \tilde{\underline{p}}^a(0)$ and $a = \max\{\underline{a}\}$.

Proof. The proof of this Lemma is given in [22].

This theorem enables us to understand the long term behavior of $\underline{p}(k)$. This theorem implies that for small fixed learning parameters for learning automata, the deviation of $\underline{p}(k)$ from $\underline{X}(.)$ over finite time interval will be made as small possible as.

Remark 2 *The interpolated process $\{\tilde{\underline{p}}^a(t)\}_{t\geq 0}$ is a sequence of random variables that takes values from $D^{m_1\times\ldots\times m_n}$, where $D^{m_1\times\ldots\times m_n}$ is the space of all functions that, at each point, are continuous on the right and have a limit on the left over $[0,\infty)$ and take values in K, which is a bounded subset of $R^{m_1\times\ldots\times m_n}$. Let $h_T(.)$ be a function over $D^{m_1\times\ldots\times m_n}$ and given by*

$$h_T(\underline{Y}) = \sup_{0\leq t\leq T} ||\underline{Y}(t) - \underline{X}(t)||,$$

for every $T<\infty$. It must be shown that with probability increasingly close to unity as $\max\{\underline{a}\}$ decreases, $\underline{p}(k)$ follows the solution of ODE (13), $\underline{X}(t)$, with an error bounded above by some fixed $\epsilon>0$. This result can be specialized to characterize the long term behavior of $\underline{p}(k)$, when the initial configuration, $\underline{p}(0)$, is in the neighborhood of an asymptotically stable compatible configuration. Let $\underline{p}^o$ be the equilibrium point to which the solution of ODE (13) when the initial condition is $\underline{p}(0)$. Using the Theorem 4, we have $\mathrm{E}\left[h_T(\tilde{\underline{p}})^a\right] \to \mathrm{E}\left[h_T(\underline{X})\right]$ as $\max\{\underline{a}\}\to 0$, where $\underline{X}$ is the solution of ODE (13). Using this and the nature of interpolation, given in (12), it is implied that for the given initial configuration and any $\epsilon>0$ and integers k_1 and k_2 $(0<k_1<k_2<\infty)$, there exists a a_0 such that

$$\mathrm{E}\left[\sup_{k_1\leq k\leq k_2} \left|\left|\underline{p}(k) - \underline{p}^o\right|\right|\right] < \epsilon \qquad \forall a < a_0,$$

where $a=\max\{\underline{a}\}$. Since $\underline{p}^o$ is an asymptotically stable equilibrium point of ODE (13), then for all initial configurations in small neighborhood of $\underline{p}^o$, the CLA converges to $\underline{p}^o$.

In the following subsections, we first find the equilibrium points of ODE (13), then study the stability property of equilibrium points of ODE (13), and finally state a theorem about the convergence of the CLA.

3.1 Equilibrium Points

The equilibrium points of equation (10) are those points that satisfy the set of equations $\Delta p_{ij}(k)=0$ for all i,j, where the expected changes in the probabilities are zero. In other words, the equilibrium points are zeros of $\underline{f}(\underline{p})$, which are studied in the following two lemmas.

Lemma 4 *All the corners of K are equilibrium points of $\underline{f}(.)$. All the other equilibrium points $\underline{p}$ of $\underline{f}(.)$ satisfy*

$$d_{iy}(\underline{p}) = d_{ir}(\underline{p}), \tag{16}$$

for all $r,y\in\{1,2,\ldots,m_i\}$, and for all $i=1,\ldots,n$.

Proof. The proof of this Lemma is given in [22].

Lemma 5 *All compatible configurations are equilibrium points of $\underline{f}(.)$.*

Proof. The proof of this Lemma is given in [22].

3.2 The Stability Property

In this subsection we characterize the stability of equilibrium configurations of CLA, that is the equilibrium points of the ODE (13). From the lemmas 4 and 5, all the equilibrium points of (13) are known. In order to study the stability of the equilibrium points of (13), the origin is transferred to the equilibrium point under consideration and then the linear approximation of the ODE is studied. The following two lemmas are concerned with the stability properties of the equilibrium points of ODE (13).

Lemma 6 *A corner $\underline{p}^o\in K^*$ is a fully compatible configuration if and only if it is uniformly asymptotically stable.*

Proof. The proof of this Lemma is given in [22].

Lemma 7 *Non-compatible equilibrium points of $\underline{f}(.)$ are unstable.*

Proof. The proof of this Lemma is given in [22].

Remark 3 *In lemmas 6 and 7, the solution of ODE (13) well characterized and it is shown that full compatibility implies uniformly asymptotic stability of the corners. In order to obtain necessary and sufficient conditions for uniformly asymptotic stability, it is essential to consider in detail the nonlinear terms in the differential equation, which appears to be a difficult problem.*

3.3 Convergence Results

We study the convergence of CLA for the following four different initial configurations, which covers all points in K.

(1) $\underline{p}(0)$ is close to a compatible corner $\underline{p}^o$. By lemma 6, there is a neighborhood around $\underline{p}^o$ entering which, the CLA will be absorbed by that corner. Thus, the CLA converges to a compatible configuration.

(2) $\underline{p}(0)$ is close to a non-compatible corner $\underline{p}^o$. By lemma 7, no matter how small neighborhood we take around $\underline{p}^o$, the solution of (13) will leave that neighborhood and enter $K-K^*$. The convergence when the initial configuration is in $K-K^*$ is discussed in case 4 below.

(3) $\underline{p}(0)\in K^*$. Using the convergence properties of L_{R-I} learning algorithm [5], no matter whether $\underline{p}(0)$ is compatible or not, the CLA will be absorbed to $\underline{p}(0)$.

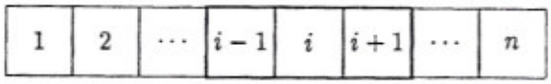


Fig. 1. The linear CLA.

(4) $\underline{p}(0) \in K - K^*$. The convergence results of the CLA for these initial configurations is stated in theorem 4.

Theorem 4 *Suppose there is a bounded differential function $D : R^{m_1+\cdots+m_n} \rightarrow R$ such that for some constant $c > 0$, $\frac{\partial D}{\partial p_{ir}}(\underline{p}) = cd_{ir}(\underline{p})$ for all i and r. Then CLA for any initial configuration in $K - K^*$ and with sufficiently small value of learning parameter $(\max\{\underline{a}\} \rightarrow 0)$, always converges to a configuration, that is stable and compatible.*

Proof. The proof of this Lemma is given in [22].

Remark 4 *If the CLA satisfies the sufficiency conditions needed for Theorem 4, then CLA will converge to a compatible configuration. When the CLA doesn't satisfy this sufficiency condition, convergence to compatible configurations cannot be guaranteed and the CLA may exhibit a limit cycle behavior [23].*

4 Synchronous Cellular Learning Automata using Commutative Rules

In this section, we study the behavior of the CLA when the commutative rules are used. Commutativity is a property of hyper matrix F^i as given in the following definition.

Definition 14 *Rule $F^i(\alpha_{i+x_1}, \ldots, \alpha_{i+x_m})$ is called commutative if and only if*

$$\begin{aligned} &F^i(\alpha_{i+x_1}, \ldots, \alpha_{i+x_m}) = \\ &F^i(\alpha_{i+x_m}, \alpha_{i+x_1}, \ldots, \alpha_{i+x_{m-1}}) = \\ &\ldots = F^i(\alpha_{i+x_2}, \alpha_{i+x_3}, \ldots, \alpha_{i+x_1}). \end{aligned} \quad (17)$$

In order to simplify the algebraic manipulations, we give the analysis is given for linear CLA. The linear CLA, as shown in figure 1, uses the neighborhood function $\vec{N}(i) = \{i-1, i, i+1\}$. The following theorem is an additional property for compatible configurations in CLA using commutative rules.

Theorem 5 *For a CLA, which uses a commutative rule, a configuration $\underline{p}$ at which $D(\underline{p})$ is a local maximum, then $\underline{p}$ is compatible.*

Proof. The proof of this Lemma is given in [22].

Remark 5 *In general, when rules of CLA are not commutative, local maxima for $D(\underline{p})$ still exist, but they may not be compatible.*

Now, using the analysis given in section 3, we can state the main theorem for the convergence of the CLA when it uses commutative rules.

Theorem 6 *A synchronous CLA, which uses uniform and commutative rule, starting from $\underline{p}(0) \in K - K^*$ and with sufficiently small value of learning parameter, $(\max\{\underline{a}\} \rightarrow 0)$, always converges to a deterministic configuration, that is stable and also compatible.*

Proof. The proof of this Lemma is given in [22].

Remark 6 *From the proof of the Theorem 6, we can conclude that the CLA converges to one of its compatible configurations, if any. If this compatible configuration is unique, then CLA converges to this configuration for which $D(\underline{p})$ is the maximum. If there are more than one compatible configurations, then the CLA depending on the initial configuration $\underline{p}(0)$ may converge to one of its compatible configurations for which $D(\underline{p})$ is a local maximum.*

Remark 7 *Theorem 6 guarantees that limit cycle for CLA does not exist and CLA always converges to an equilibrium of ODE.*

5 Numerical Examples

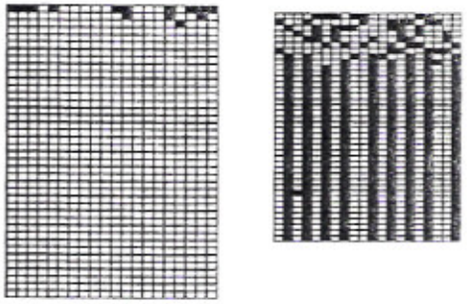
This section discusses patterns formed by the evolution of cellular learning automata from random initial configuration, chosen by the learning automata in cells. Different cellular learning automata rules are found to yield different configurations. For the sake of simplicity in our presentation, we use the following notation to specify the rules for CLA for which each cell has a learning automaton with m actions. The actions of each learning automaton are represented by integers in interval $[0, m-1]$. Hence, the configuration of each cell and its neighbors forms a $\bar{m}$ digits number in interval $[0, m^{\bar{m}} - 1]$ with $m^{\bar{m}}$ possible values. The value of reinforcement signal for all of the above $m^{\bar{m}}$ configurations constitute an $m^{\bar{m}}$ bit number. Then the rule identified by decimal representation of this $m^{\bar{m}}$-bit number. For the sake of simplicity in our presentation, we use notation $(j)_m$ to specify the rules in the CLA, where j is a decimal representing the rule and m is the number of actions for each learning automaton. For example, the following table represents the rule 22 for a linear CLA with two-actions learning automata and represented by $(22)_2$. Each of the eight possible sets configuration for a cell and its neighbors appear on the upper row, while the lower row gives the value of the reinforcement signal to be taken to the central cell on the next time step.

In the experiments presented below, the synchronous CLA are considered. Figures 2 through 6, show the time-space diagram evolution of CLA using commutative rules with 20 cells and a two-actions L_{R-I} learning automaton in each cell.

Figure 7, shows the time-space diagram evolution of CLA using noncommutative rules with 20 cells and a two-actions L_{R-I} learning automaton in each cell.

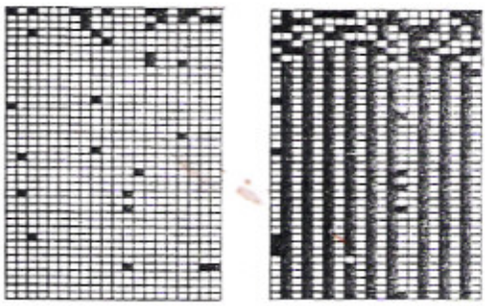
Table 1:
The scheme for the rule numbering for two actions LA.

Configuration	Reinforcement Signal
111	0
110	0
101	0
100	1
011	0
010	1
001	1
000	0



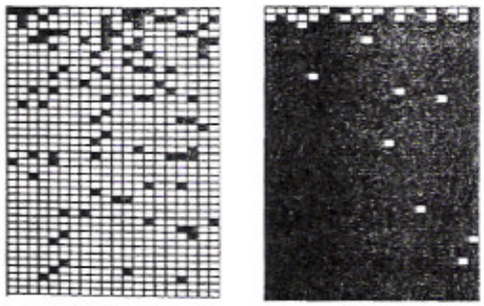
rule $(1)_2$ rule $(22)_2$

Fig. 2. Time-space diagram of synchronous CLA using commutative rules



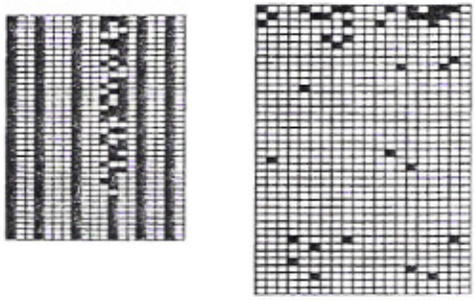
rule $(23)_2$ rule $(126)_2$

Fig. 3. Time-space diagram of synchronous CLA using commutative rules



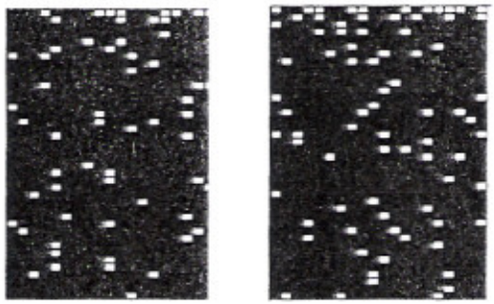
rule $(127)_2$ rule $(128)_2$

Fig. 4. Time-space diagram of synchronous CLA using commutative rules



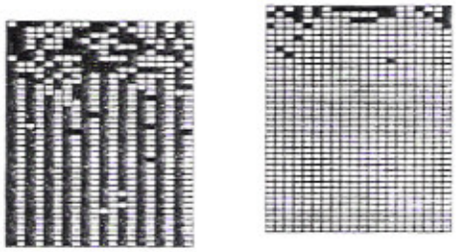
rule $(150)_2$ rule $(151)_2$

Fig. 5. Time-space diagram of synchronous CLA using commutative rules



rule $(232)_2$ rule $(233)_2$

Fig. 6. Time-space diagram of synchronous CLA using commutative rules



rule $(180)_2$ rule $(190)_2$

Fig. 7. Time-space diagram for CLA

The simulation results show that the CLA converges to a configuration in K^* rather than to a configuration in $K - K^*$.

6 Conclusions

In this paper, the formal description of cellular learning automata is given and its convergence behavior is studied. It is shown that for commutative rules, the cellular learning automata converges to a stable configuration for which the average rewards for the CLA is maximized. The numerical results also confirms the theory.

It is apparent that admissible compatible is the most interesting configuration to be searched. However, the $CLA(L_{R-I})$ using commutative rules converges only to one of compatible configurations which are the local maxima of D. Hence,

$CLA(L_{R-I})$ cannot always converge to admissible configuration. The convergence to admissible compatible configuration is possible if we allow some information exchange, in addition to the information exchange for calculating the reinforcement signal, among the learning automata.

References

[1] E. Fredkin, "Digital Machine: A Informational Process Based on Reversible Cellular Automata," *Physica D*, vol. 45, pp. 254–270, 1990.

[2] N. H. Packard and S. Wolfram, "Two-Dimensional Cellular Automata," *Journal of Statistical Physics*, vol. 38, pp. 901–946, 1985.

[3] M. Mitchell, "Computation in Cellular Automata: A Selected Review," tech. rep., Santa Fe Institute, Santa Fe, NM, USA, Sept. 1996.

[4] J. Kari, "Reversability of 2D Cellular Automata is Undecidable," *Physica D*, vol. 45, pp. 379–385, 1990.

[5] K. S. Narendra and K. S. Thathachar, *Learning Automata: An Introduction*. New York: Printice-Hall, 1989.

[6] P. R. Srikantakumar and K. S. Narendra, "A Learning Model for Routing in Telephone Networks," *SIAM Journal of Control and Optimization*, vol. 20, pp. 34–57, Jan. 1982.

[7] B. J. Oommen and E. V. de St. Croix, "Graph Partitioning Using Learning Automata," *IEEE Transactions on Commputers*, vol. 45, pp. 195–208, Feb. 1996.

[8] B. J. Oommen and T. D. Roberts, "Continuous Learning Automata Solutions to the Capacity Assignment Problem," *IEEE Transactions on Commputers*, vol. 49, pp. 608–620, June 2000.

[9] M. R. Meybodi and H. Beigy, "A Note on Learning Automata Based Schemes for Adaptation of BP Parameters ," *Journal of Neuro Computing*, vol. 48, pp. 957–974, Nov. 2002.

[10] M. R. Meybodi and H. Beigy, "New Learning Automata Based Algorithms for Adaptation of Backpropagation Algorithm Parameters," *International Journal of Neural Systems*, vol. 12, pp. 45–68, Feb. 2002.

[11] K. Krishna, "Cellular Learning Automata: A Stochastic Model for Adaptive Controllers," Master's thesis, Department of Electrical Engineering, Indian Institue of Science, Banglore, India, June 1993.

[12] M. R. Meybodi, H. Beigy, and M. Taherkhani, "Cellular Learning Automata and its Applications," *Accepted for Publication in Journal of Sharif*.

[13] M. R. Meybodi, H. Beigy, and M. Taherkhani, "Application of Cellular Learning Automata to Image Processing," in *Proceedings of First Conference in Mathematics and Communication, Iranian Telecommunication Research Center, Tehran, Iran*, pp. 23.1–23.10, Oct. 2000.

[14] M. R. Meybodi and M. Taherkhani, "Application of Cellular Learning Automata in Modeling of Rumor Diffusion," in *Proceedings of 9th Conference on Electrical Engineering, Power and Water Institute of Technology, Tehran, Iran*, pp. 102–110, May 2001.

[15] M. R. Meybodi and M. R. Kharazmi, "Image Restoration Using Cellular Learning Automata," in *Proceedings of the First Iranian Conf. on Machine Vision, Image Processeing and Applications, Birjand, Iran*, pp. 244–254, May 2000.

[16] M. R. Kharazmi and M. R. Meybodi, "Image Segmentation Using Cellular Learning Automata," in *Proceedings of 10th Iranian Conference on Electrical Engineering, ICEE-96, Tabriz, Iran*, May 2001.

[17] M. R. Kharazmi and M. R. Meybodi, "Image Restoration Using Cellular Learning Automata," in *Proceedings of the Second Iranian Conf. on Machine Vision, Image Processeing and Applications, Tehran, Iran*, pp. 261–270, 2003.

[18] M. R. Meybodi and M. R. Kharazmi, "Application of Cellular Learning Automata to Image Processing," *Journal of Amirkabir, to be published*, 2003.

[19] M. R. Meybodi and M. R. Khojasteh, "Application of Cellular Learning Automata in Modelling of Commerce Networks," in *Proceedings of 6th Annual International Computer Society of Iran Computer Conference CSICC-2001, Isfehan, Iran*, pp. 284–295, Feb. 2001.

[20] H. Beigy and M. R. Meybodi, *A Self-Organizing Channel Assignment Algorithm: A Cellular Learning Automata Approach*, vol. 2690 of *Springer-Verlag Lecture Notes in Computer Science*, pp. 119–126. Springer-Verlag, 2003.

[21] M. R. Meybodi and F. Mehdipour, "VLSI Placement Using Cellular Learning Automata," in *Proceedings of 8th Annual International Computer Society of Iran Computer Conference CSICC-2001, Mashad, Iran*, pp. 195–203, Mar. 2003.

[22] H. Beigy and M. R. Meybodi, "A Mathematical Framework to Study the Evolution of Cellular Learning Automata," Tech. Rep. TR-CE-2003-004, Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran, 2003.

[23] K. S. Narendra and A. M. Annaswarmy, *Stable Adaptive Systems*. New York: Printice-Hall International Inc., 1989.

انجمن کامپیوتر ایران
Computer Society of Iran

مجموعه مقالات

(جلددوم)

(مشتمل بر مقالات انگلیسی و مقالات پوستری)

نهمین

کنفرانس سالانه

انجمن کامپیوتر ایران

۲۸ تا ۳۰ بهمن ماه ۱۳۸۲
دانشگاه صنعتی شریف

ITRC

مرکز
فناوری
اطلاعات
و ارتباطات
پیشرفته