Error Correcting Capability of Cellular Automata Based Associative Memory

Pradipta Maji, Niloy Ganguly, and P. Pal Chaudhuri, *Senior Member, IEEE*

***Abstract*—This paper reports the error correcting capability of an associative memory model built around the sparse network of cellular automata (CA). Analytical formulation supported by experimental results has demonstrated the capability of CA based sparse network to memorize unbiased patterns while accommodating noise. The desired CA are evolved with an efficient formulation of simulated annealing (SA) program. The simple, regular, modular, and cascadable structure of CA based associative memory suits ideally for design of low cost high speed online pattern recognizing machine with the currently available VLSI technology.**

***Index Terms*—Associative memory, cellular automata (CA), generalized multiple attractor CA (GMACA), multiple attractor CA (MACA), pattern recognition, simulated annealing (SA).**

I. Introduction

THIS paper reports an associative memory model to address the problem of "pattern recognition." The solution is based on an elegant computing model of a particular class of sparse network referred to as cellular automata (CA). Analytical formulation supported with extensive experimental results has established CA based associative memory as an efficient and cost-effective alternative to the solutions generated with dense network of neural net.

By convention, the task of pattern recognition demands automatic identification of objects and images by their shapes, forms, outlines or some other attributes. In the internetworked society of cyber-age, the task of pattern recognition has become ubiquitous, involving automated recognition of specific patterns of bit/symbol string. It is an integral part in machine intelligence systems.

In conventional approach of pattern recognition, machine compares the given input pattern with each of the learnt/stored patterns and identifies the closest match. If there are k patterns, time to recognize the closest match is $O(k)$. As the number of stored patterns goes up, recognition process becomes slow.

Over the years, research in the field has been directed toward development of recognition algorithm in constant time. In pattern recognition, there are mainly two steps— 1) learning the pattern set; and 2) identifying one of the learnt patterns that has closest match to the given input pattern to be recognized. The process of learning a pattern set involves one time computation cost, while recognition of the incoming patterns is a repetitive process. In this context, schemes which take longer time for learning but reduce delay of the recognition process, have attracted considerable attention.

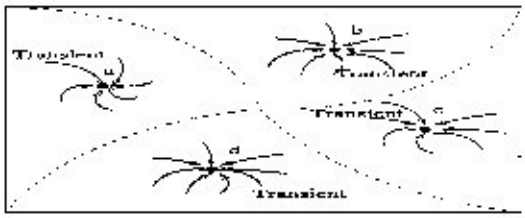


Fig. 1. Model of associative memory.

The "associative memory" model provides a solution to the problem where time to recognize a pattern is independent of the number of patterns learnt/stored [1]. This model divides the entire state space into pivotal points a-d (Fig. 1) that represent the patterns learnt. The pivotal points are also referred to as "stable states." On the other hand, states close to a pivotal point get associated with the corresponding learnt pattern and are referred to as "transient states." A transient state can be also viewed as a stable state with some noise. Identification of an input pattern (without or with distortion due to noise) amounts to traversing the transient path, as shown in Fig. 1, from the given input pattern to the closest pivotal point. As a result, the process of recognition "associates" a set of transient states with a learnt pattern and the process of identification becomes independent of the number of patterns learnt. A logical evolution is to build the associative memory model for pattern recognition.

In early 1980s, the seminal work of Hopfield [9] made a breakthrough by modeling a recurrent, asynchronous, neural net as an associative memory system. Extensive research [2], [10], [13], [16], [22] have been reported in this field in last two decades. However, the user community of the networked society of present age look forward for high speed, low cost, online pattern recognition systems. The dense network of neural net and its complex structure cannot efficiently support such demands of current age. In this background, search for alternative model of pattern recognizing machine around the simple sparse network of CA continued [3], [11], [17], [18], [20]. The simple, regular, modular, cascadable local neighborhood structure of CA serves as an excellent sparse network. This network, as reported in [4], can be efficiently realized in hardware.

In this paper, we propose an elegant model of associative memory for pattern recognition. The model is built around a general class of CA termed as generalized multiple attractor CA

Manuscript received June 11, 2002; revised October 31, 2002. This paper was recommended by Associate Editor Y. Pan.

P. Maji and P. P. Chaudhuri are with the Department of Computer Science and Technology, Bengal Engineering College (DU), Howrah, 711103 India (e-mail: pradipta@cs.becs.ac.in, ppc@cs.becs.ac.in).

Niloy Ganguly is with the IISWBM, Calcutta, 700073 India (e-mail: n_ganguly@ hotmail.com).

(GMACA). This specific class of CA displays encouraging results in the field of pattern recognition [5], [6], [8], [14], [15]. While in [8] we have mainly dealt with memorizing capacity of CA based associative memory, in the present one we focus our attention on its error correcting capability. The evolutionary scheme of simulated annealing (SA) is employed to evolve desired GMACA configurations.

In the above context we present CA preliminaries in Section II followed by an overview of GMACA synthesis scheme in Section III. Theoretical analysis of the error correcting capability of GMACA rule space is reported in Section IV. The problem of exponential complexity of synthesis algorithm has been addressed with efficient formulation of simulated annealing (SA) in Section V. The rule space theoretically derived in Section IV, can be found to be a subset of the CA rules of GMACA evolved with SA program. Finally, the experimental results reported in Section VI confirm the GMACA based associative memory as an efficient model for pattern recognition.

II. Cellular Automata (CA)

A CA consists of a number of cells organized in the form of a lattice. It evolves in discrete space and time [19]. The next state of a cell depends on its own state and the states of its neighboring cells. In this paper we develop an associative memory model around three-neighborhood (left, self, and right neighbor) one-dimensional (1–D) CA with each cell having two states—0 or 1.

In a two state three-neighborhood CA, there can be a total of 2^{2^3}—that is, 256 distinct next state functions. If the next state function of a cell is expressed in the form of a truth table, then the decimal equivalent of the output is referred to as "rule" for the cell [23]. Two sample rules 90 and 150 are illustrated in the equation at bottom of the page.

The first row lists the possible 2^3—that is, eight combinations of present states (left, self, and right) for a three-neighborhood CA cell at time t. The next two rows list the next states for the ith cell at time instant $(t+1)$ for two different rules. The truth tables specified on the rows provide the next state logic of CA rules. For example

$$\text{Rule 90: } q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$$
$$\text{Rule 150: } q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$$

where $\oplus$ denotes XOR logic function.

Out of total 256 rules there are only 14 rules that employ linear Additive logic of XOR/XNOR function. Such CAs are referred to as Linear/Additive CA [4]. Other rules employ nonlinear logic involving AND, OR, NOT, etc., logic functions. An n-cell hybrid CA (with different rules applied to different cells) is configured with the rule vector $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_n \rangle$ where ith cell is configured with the rule $\mathcal{R}_i$; each $\mathcal{R}_i$ being one of the possible 256 rules. In an uniform CA, same rule is applied on each of the CA cells.

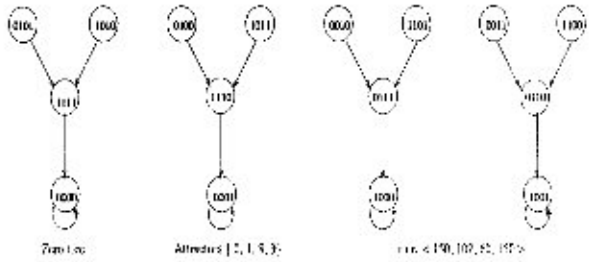


Fig. 2. State space of a four-cell MACA divided into four attractor basins.

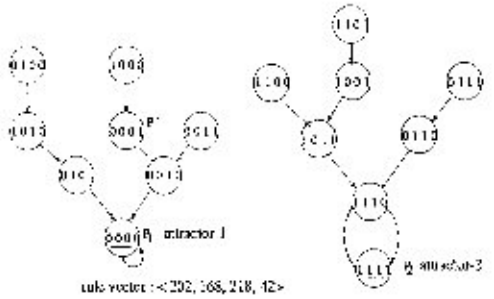


Fig. 3. State space of a GMACA based 4-bit pattern recognizer with rule vector $\langle 202, 168, 218, 42 \rangle$.

A. Generalized Multiple Attractor CA

The concept of Multiple Attractor Cellular Automata (MACA) has been introduced in [4]. Its state transition behavior consists of multiple components—each component, as noted in Fig. 2, is an inverted tree. A node with self loop is referred to as an attractor. The nodes with binary number 0000, 0001, 1000, and 1001 are the attractors of the four components in the MACA of Fig. 2. The set of nodes in a component with attractor α is referred to as α-basin.

Even though such an MACA displays interesting characteristics, it employs only XOR/XNOR rules. This rule set is functionally incomplete in the sense that any next state function cannot be realized with this restricted rule set. Hence, the research reported in this paper explores a more general class of MACA termed as Generalized Multiple Attractor CA (GMACA) designed with nonlinear CA rules implementing all possible logic functions—AND, OR, XOR, NOT, etc.

A GMACA is a hybrid CA—that is, unlike uniform CA, same rule is not applied for each of the cells. It can efficiently model an associative memory [5]–[8], [14], [15] to perform pattern recognition task. Fig. 3 illustrates the state space of a four-cell hybrid GMACA with rule vector $\langle 202, 168, 218, 42 \rangle$—that is, rule 202 is applied on left most cell, followed by rule 168 on next one and so on. The state space of this CA is divided into two

Neighborhood	111	110	101	100	011	010	001	000	Rule
(i) Next State	0	1	0	1	1	0	1	0	90
(ii) Next State	1	0	0	1	0	1	1	0	150

attractor basins—Basin-1 and Basin-2 built around attractor-1 and attractor-2. Unlike MACA (Fig. 2), a GMACA has attractors of cycle length greater than or equal to 1. For the GMACA of Fig. 3, the attractor-1 and attractor-2 have cycle length 1 and 2, respectively. The states in a basin not covered by the attractor cycles are referred to as "transient states" in the sense that a CA finally settles down in one of its attractor cycles after passing through such transient states.

B. GMACA Modeling an Associative Memory

A GMACA having its state space distributed into disjoint basins (Fig. 3) with transient and attractor states, models an associative memory. The patterns to be learnt, as shown in Fig. 1, are modeled as "pivotal point/stable state." The entire state space built around the given set of pivotal points is the state space generated by a GMACA with its attractors and transient states. Memorizing the set of pivotal points $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots \mathcal{P}_k\}$ is equivalent to design of a GMACA with the pivotal points as the states in different attractor cycles. The states in an attractor cycle other than the pivotal point are viewed as "pseudo-transient states/points" in the present work and treated as transient states. Any transient point $\acute{\mathcal{P}}_i$, with limited distance from a pivotal point $\mathcal{P}_i$, can be considered as a pattern with some noise. The correct output $\mathcal{P}_i$ can be produced in time proportionate to the traversal of GMACA from the state $\acute{\mathcal{P}}_i$. In the present work we have assumed hamming distance (HD) as the distance metric.

A GMACA satisfying following two relations models an associative memory storing the set $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots \mathcal{P}_k\}$.

1) **R1:** Each attractor basin of the GMACA should contain one and only one pattern $(\mathcal{P}_i)$ to be learnt in its attractor cycle; the corresponding basin is referred to as $\mathcal{P}_i$-basin.
2) **R2:** The hamming distance (HD) of each state $\acute{\mathcal{P}}_i \in \mathcal{P}_i$-basin with $\mathcal{P}_i$ is lesser than that of $\acute{\mathcal{P}}_i$ with any other $\mathcal{P}$'s,—that is, $\mathrm{HD}(\acute{\mathcal{P}}_i, \mathcal{P}_i) < \mathrm{HD}(\acute{\mathcal{P}}_i, \mathcal{P}_j), \mathcal{P}_j \in \mathcal{P}$ and $\mathcal{P}_j \neq \mathcal{P}_i$.

The relation **R1** ensures uniqueness of the stored patterns in an attractor cycle, whereas the relation **R2** ensures recognition of patterns with distortion due to noise. The following example illustrates the relations for the example GMACA of Fig. 3.

Example 1: To design a machine for recognition of two patterns $\mathcal{P}_1 = 0000$ and $\mathcal{P}_2 = 1111$ with single bit noise, we first synthesize a CA (rule vector) for which the state transition behavior of GMACA is similar to that of Fig. 3 while satisfying both the relations **R1** and **R2**.

It learns two patterns, $\mathcal{P}_1$ 0000 and $\mathcal{P}_2$ 1111. The state $\acute{\mathcal{P}} = 0001$ has the hamming distances 1 and 3 with $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Let $\acute{\mathcal{P}}$ be given as the input and its closest match is to be identified with one of the learnt patterns. The recognizer designed with the GMACA of Fig. 3 is loaded with $\acute{\mathcal{P}} = 0001$ and returns the desired pattern $\mathcal{P}_1$ after two time steps.

Next section addresses the problem of GMACA synthesis scheme for recognizing a given set of patterns.

III. GMACA Synthesis Scheme

The synthesis algorithm of GMACA can be viewed as the training phase of CA based pattern recognizer to recognize a given set of patterns. The output of synthesis algorithm is GMACA based pattern recognizer with its rule vector specifying the rule number applied for each of the cells. For example, the rule vector of the GMACA having pattern recognizing capability illustrated in Example 1 is $\langle 202, 168, 218, 12 \rangle$.

A. Design of GMACA Rule Vector

The GMACA synthesis algorithm consists of three phases. It is assumed to recognize patterns with maximum noise of $r_{\max}$ bits.

1) *Phase I*—Generate a directed graph with a cycle, and map a pattern $\mathcal{P}_i$ (to be learnt) on to a cyclic node of this graph. Also, map the noisy patterns $\acute{\mathcal{P}}_i$ (where $\mathrm{HD}(\mathcal{P}_i, \acute{\mathcal{P}}_i) \leq r_{\max}$) to other nodes of the graph. Two associated parameters—number of nodes in a graph and its cycle length—have been dealt with in subsequent discussions. In order to recognize k patterns, k number of directed graphs are generated.
2) *Phase II*—Derive state transition table for each of the graphs generated in *Phase I*.
3) *Phase III*—Generate rule vector of the GMACA satisfying the next state logic specified in the state transition tables derived in *Phase II*.

The synthesis algorithm accepts the pattern set $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots \mathcal{P}_k\}$ of cardinality k to be learnt as the input, each $\mathcal{P}_i (i = 1, 2, \ldots, k)$ being an n-bit pattern. It cycles through three phases in successive stages till desired n-bit GMACA with k-attractor basins is reached or the specified time limit gets elapsed with null output. The search is guided by the rules **R1** and **R2** reported in Section II.B. The rule **R1** ensures presence of only one pattern $\mathcal{P}_i \in \mathcal{P}$ in the cycle of a graph that can be viewed as an attractor cycle of a GMACA to be synthesized. "Maximum permissible cycle length" is assumed to be $l_{\max}$. For implementing **R2**, the algorithm accepts an input $r_{\max}$ that specifies "maximum permissible noise." That is, a noisy pattern say $\acute{\mathcal{P}}_i$ (where $\mathrm{HD}(\mathcal{P}_i, \acute{\mathcal{P}}_i) = r_{\max}$) should be correctly recognized as $\mathcal{P}_i$ by GMACA based model.

The three phases of synthesis algorithm is next illustrated with an example.

Phase I—Since, the state transition diagram of a GMACA can be conceived as a graph, we first randomly generate k number of directed graphs with each node coded as an n-bit string. The cycle length l permissible for the generated graph is assumed to be less than or equal to $l_{\max}$. Each graph represents a basin of the candidate GMACA to be synthesized, while its cycle represents an attractor cycle. Each directed graph has a unique pattern $\mathcal{P}_i$ in its attractor cycle, while those with limited noise added to $\mathcal{P}_i$ are the patterns in the $\mathcal{P}_i$-basin.

The number of nodes p of each graph is equal to the number of states in the corresponding basin—that is, the patterns without or with specified noise. So

$$p = \sum_{r=0}^{r_{\max}} \binom{n}{r} \tag{1}$$

where r is the number of noisy bits and $r_{\max}$ is the maximum permissible noise that can be tolerated by GMACA based pattern recognizer. After mapping a particular pattern (say $\mathcal{P}_i$ to be

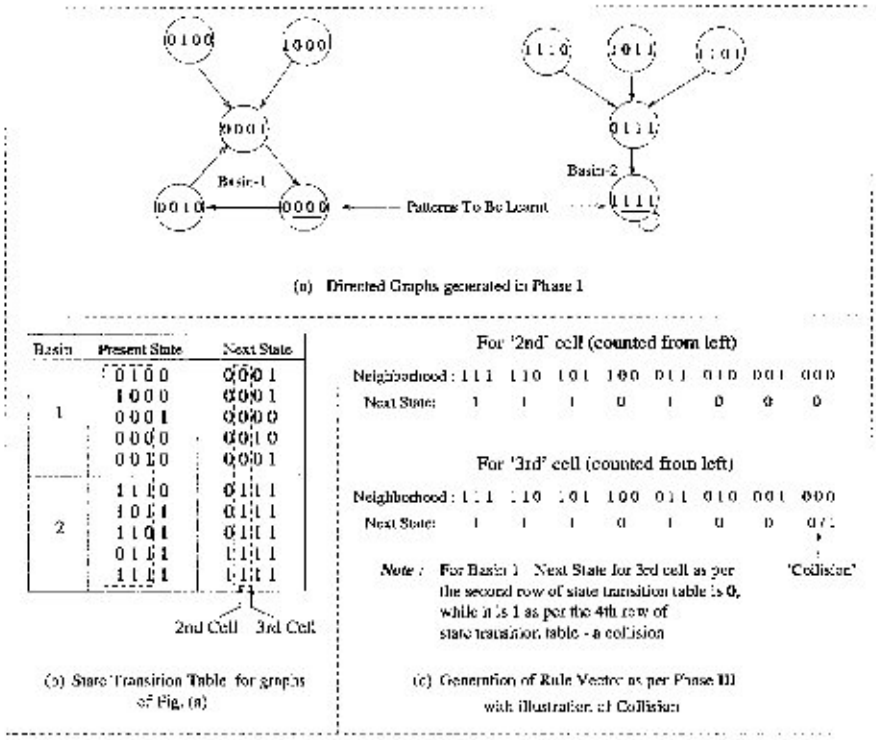


Fig. 4. Randomly generated directed graphs with state transition tables and CA rules.

learnt) in the cyclic node of a graph, we randomly map other patterns to be covered by $\mathcal{P}_i$-basin at different nodes of same graph. Note that the $\mathcal{P}_i$-basin of GMACA to be synthesized covers the states with permissible noise of r bits ($r = 0$ to $r_{\max}$) added to $\mathcal{P}_i$. A design example follows.

Example 2: Fig. 4(a) represents two arbitrary graphs generated in *Phase I* for $n = 4$ and $r_{\max} = 1$. Patterns to be learnt $\mathcal{P}_1$ 0000 and $\mathcal{P}_2$ 1111 are mapped onto the nodes of the attractor cycle of length 3 and 1 respectively. The sets 0001, 0010, 0100, 1000 and 1110, 1101, 1011, 0111 are the noisy patterns with noise of 1 bit added to $\mathcal{P}_i (i = 1, 2)$ respectively. These are mapped in two attractor basins as shown in Fig. 4(a).

Phase II—Fig. 4(b) represents a state transition table derived from two directed graphs shown in Fig. 4(a). Total number entries in the state transition table is $(k \cdot p)$, where k is the number of patterns to be learnt and p is the number of states in each graph (1). For the example graphs of Fig. 4(a), the number of patterns to be learnt (k) is 2 ($\mathcal{P}_1$ and $\mathcal{P}_2$) and the number of states in each basin (p) is 5. So, total number entries in the state transition table of Fig. 4(b) is 10.

Phase III—Design rule vector of the GMACA cells from the state transition table.

Consider a cell (say ith cell) whose rule is to be identified. We concentrate on three columns—$(i-1)$th, ith and $(i+1)$th columns of all the $(k \cdot p)$ number of patterns of the state transition table of Fig. 4(b). Suppose, for a present state configuration of the ith cell, the next state is '0' for n_0 times and "1" for n_1 times, then state "0" and "1" collides with each other to become the next state of the ith cell for that configuration.

Fig. 4(c) represents the neighborhood configurations along with the next state of second and third cells of the patterns of two basins noted in Fig. 4(b). For second cell, there is no collision between state "0" and "1" for eight possible configurations. Whereas for "000" neighborhood configuration, the next state of 3rd cell is "0" (2nd row of Fig. 4(b)) for 1 time and "1" [fourth row of Fig. 4(b)] for one time—that is, $n_0 = 1$ and $n_1 = 1$. So, for "000" configuration, the next state of third cell may be "0" or "1" generating an instance of collision.

In order to resolve this conflict we introduce the following heuristic.

B. Resolution of Collision

1) If $n_0 \simeq n_1$, the collision between state "0" and "1" is high. In that case, we randomly decide the next state of a cell.
2) If $n_0 \gg n_1$ or $n_0 \ll n_1$, the collision is minimum. In that case, the next state of a cell is '0' if $n_0 > n_1$, otherwise "1."

The synthesis scheme searches for desired GMACA conforming rules **R1** and **R2** while resolving the collision, if there is any. Characterization of GMACA rule space undertaken in next section aims to reduce this search space.

IV. Characterization of Rule Space of Desired GMACA

This section characterizes the rule space of desired GMACA displaying pattern recognition capability. Such characterization is based on noise immunity of synthesized GMACA to recognize patterns.

The characterization of rule space proceeds under the following assumptions.

Assumptions:

1) Attractor cycles of GMACA are of length 1.

2) GMACA is designed to recognize patterns with single bit noise.

   Note: (According to the analysis reported in Section VI-A1, this does not pose any restriction to recognize patterns corrupted with multiple bit noise).
3) Transient length of the pattern corrupted with single bit noise is equal to 1.
4) Rule space generated for a GMACA designed to recognize two patterns is a superset of the rule space generated to recognize more than two patterns.

Each of the above assumptions, as per the analysis and experimental results reported in Sections VI-A1, VI-A2, VI-B1, VI-C, can be found to be valid for the GMACA model developed for pattern recognition.

Based on the above assumptions, the pattern recognizer is built with three-neighborhood CA. So in order to analyze the noise immunity (that is, error correcting capability (ECC)) of the ith cell, we concentrate only on the bit patterns of $(i-1)$th, ith and $(i-1)$th cells as explained below.

Definition 1: For k number of n-bit patterns, total number of noisy patterns with single bit noise is $(k \cdot n)$ [as per the Assumption (2)]. To ensure ECC for the ith cell, its rule should be so designed that the cell recovers correct bit irrespective of the noise on $(i-1)$th, ith, or $(i-1)$th bit positions [as per the Assumptions (3)]. So, total number of noisy patterns with single bit noise to be recovered by the rule of ith cell is $(3 \cdot k)$. Thus, the error correcting capability (ECC) of the ith cell is defined as the ratio of number of noisy patterns correctly recognized and $(3 \cdot k)$.

For ease of subsequent discussions we introduce the following terminologies. All three bit configurations are noted as a bit string of $b_{i-1}\ b_i\ b_{i+1}$ with middle one refers to ith cell (Fig. 5).

Terminologies:

1) $\mathcal{P}_{\text{same}}$ denotes the probability that the state of ith cell of all attractors is same.
2) $\mathcal{P}_{\text{diff}}$ denotes the probability that the state of ith cell of all attractors is different.
3) $\text{HD}(***, ***)$ denotes the hamming distance between 3-bit neighborhood configurations of the ith cell of two attractors—that is, $\text{HD}(b_{i-1}\ b_i\ b_{i+1}, \acute{b}_{i-1}\ \acute{b}_i\ \acute{b}_{i+1})$ (Fig. 5).
4) m denotes the number of distinct neighborhood configurations of the ith cell $(i = 1, 2, \ldots)$; hence $m \leq 8$.
5) $\text{PO}(m)$ denotes the probability of occurrence of m distinct neighborhood configurations for the ith cell in the patterns to be learnt.
6) $\mathcal{N}_j$ denotes the number of configuration pairs of ith cell with HD as j.

Fig. 5. Sixteen possible configurations with b_{i-1}, b_{i+1} and $\acute{b}_{i-1}, \acute{b}_{i+1}$; $b_i \neq \acute{b}_i$.

7) ECC(1) denotes the error correcting capability of the ith cell if state of the ith cell is same in the patterns to be learnt.
8) $\text{ECC}(m)$ denotes the error correcting capability of the ith cell with m number of distinct neighborhood configurations in the patterns to be learnt.
9) ECC_j denotes the error correcting capability when HD between two configurations is j.

Based on the Assumption (4), we first characterize CA rule space for "two attractor basins" GMACA recognizing two patterns.

A. Rule Space of GMACA With Two Attractor Basins

Under the assumption of single cycle attractor [Assumption (1)], we investigate effective CA rules for "two attractor basins" GMACA recognizing two patterns without or with single bit noise. The CA rule corresponding to the ith cell depends on its next state value for different neighborhood configurations in the GMACA state transition graph. For example, in Fig. 5, the CA rule corresponding to the second cell depends on its next state value for different neighborhood configurations (i.e., $111, 110, \ldots, 000$). If the state of the ith cell is different, there are 2^3—that is, 16 possibilities (Fig. 5). Evaluation of each of these cases are next formalized in the following theorems based on the analysis of bit patterns of two attractor states.

Theorem 1: If state of the ith cell of two attractors is same, its error correcting capability (ECC(1)) is 100%.

Proof: Suppose, the state of the ith cell is 0 for two attractors. The neighborhood configurations of the ith cell may be 000, 001, 100, 101, which must map to 0. Besides these, as the cell accommodates some noise, other neighborhood configurations (i.e., 010, 011, 110, 111) which are regarded as the noisy configurations of the ith cell corrupted with single or multiple bit noise, also map to 0. So, there is no collision between state "0" and "1" for any one of eight configurations; thus, it can recover single bit noise at one of the three positions—$(i-1), i, (i+1)$. In this case, the effective rule of the ith cell is 0 and the error correcting capability is 100%. Similarly, if the state of the ith cell is 1, the effective rule must be 255 to act as an useful memory and its error correcting capability is 100%. Two rules 0 and 255 are illustrated in the equation at the bottom of the page. ■

Neighborhood	111	110	101	100	011	010	001	000	Rule
(i) Next State:	0	0	0	0	0	0	0	0	0
(ii)Next State	1	1	1	1	1	1	1	1	255

CA Rules: If the state of the ith cell of two attractors is same, the effective CA rule is 0 if the state is 0, otherwise 255.

Theorem 2: If $\mathrm{HD}(*0*, *1*) = 1$, the error correcting capability (ECC_1) of the ith cell is 66.67%, where $*$ denotes any bit 0 or 1.

Proof: If $\mathrm{HD}(*0*, *1*) = 1$, two configurations of two attractors of the ith cell differ only at their ith position; $(i-1)$th and $(i+1)$th positions of two configurations are same. So, two noisy configurations corrupted with single bit noise at ith position create a collision while that of $(i-1)$th and $(i+1)$th positions don't create any collision. So, the ith cell can recover noise at $(i-1)$th and $(i-1)$th positions but not at ith position; thus the error correcting capability of the ith cell is $(2 \cdot k/3 \cdot k)$—that is, 66.67%. ■

CA Rules: If $\mathrm{HD}(*0*, *1*) = 1$, there are 4 possible configurations (N_1) of the ith cell—(000 010), (001 011), (100 110) and (101 111). In these cases, the effective CA rules are Rule 76, 108, 140, 156, 196, 198, 200, 201, 204, 205, 206, 220 and 236.

Theorem 3: If $\mathrm{HD}(*0*, *1*) = 2$, the error correcting capability (ECC_2) of the ith cell is 66.67%.

Proof: If $\mathrm{HD}(*0*, *1*) = 2$, two configurations of the ith cell differ at their ith position as well as one of the $(i-1)$th or $(i+1)$th positions. So, the ith cell can accommodate noise on ith position and one of its $(i-1)$th or $(i+1)$th positions where the state is same. But, it cannot recover noise of third positions (where the state is different) as there are collisions between state "0" and "1." Thus, the error correcting capability of the ith cell is $(2 \cdot k/3 \cdot k)$—that is, 66.67%. Hence, the result follows. ■

CA Rules: If $\mathrm{HD}(*0*, *1*) = 2$, there are eight possible configurations (N_2) of the ith cell. These configurations are (000, 011), (100, 111), (000, 110), (001, 111), (001, 010), (101, 110), (100, 010) and (101, 011). The effective CA rules for these configurations are Rule 12, 13, 14, 15, 68, 69, 76, 77, 78, 79, 84, 85, 92, 93, 136, 138, 140, 141, 142, 143, 168, 170, 172, 174, 192, 196, 197, 200, 202, 204, 205, 206, 207, 208, 212, 213, 216, 220, 221, 224, 228, 232, 234, 236, 238, 240, 244, 248, and 252.

Theorem 4: If $\mathrm{HD}(*0*, *1*) = 3$, the error correcting capability (ECC_3) of the ith cell is 100%.

Proof: If $\mathrm{HD}(*0*, *1*) = 3$, two configurations of two attractors of the ith cell differ at all their three positions. So, the hamming distance between one of noisy configurations corrupted with single bit noise at $(i-1)$th, ith or $(i+1)$th positions of an attractor with attractor itself is always less than that of another attractor; in effect there is no collision between state "0" and "1" for any one of eight possible present state configurations. Thus the ith cell can recognize a noisy configuration correctly if 1 out of 3 bits differs from its original configuration. Hence, the error correcting capability of the ith cell is 100%. ■

CA Rules: If $\mathrm{HD}(*0*, *1*) = 3$, there are four possible configurations of the ith cell. These are (000, 111), (001, 110), (010, 101) and (100, 011). The effective CA rules for these configurations are Rule 77, 142, 212, and 232.

Theorem 5: If the state of the ith cell of "two attractors" is different, the error correcting capability (ECC(2)) of the ith cell is 75%.

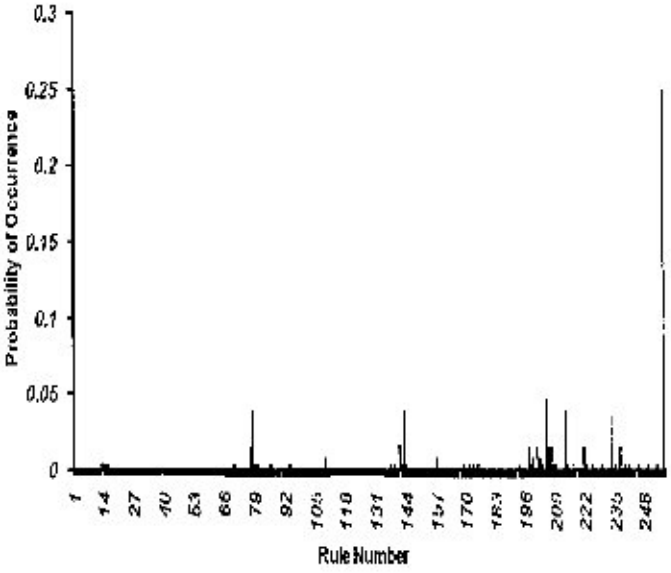


Fig. 6. Distribution of CA rules for two attractor basins.

Proof: If the state of the ith cell is different for "two attractors," then according to Theorems 2–4, the error correcting capability [ECC(2)] is given by

$$\mathrm{ECC}(2) \quad \frac{\sum_{j=1}^{3} \mathrm{ECC}_j \cdot N_j}{\sum_{j=1}^{3} N_j} \quad 75\%. \tag{2}$$

Hence, the result follows. ■

Theorem 6: The average error correcting capability of the ith cell of "two attractor basins" GMACA to recover single bit noise in single time step is 87.5%.

Proof: The probability that the state of the ith cell of two attractors is same ($\mathcal{P}_{\mathrm{same}}$) is 0.5 and that of different ($\mathcal{P}_{\mathrm{diff}}$) is 0.5. Now, according to Theorem 1 and Theorem 5, if the ith cell of "two attractor basins" GMACA recovers single bit noise in single time step, $\mathrm{ECC}(1) = 100\%$ and $\mathrm{ECC}(2) = 75\%$. So, the average error correcting capability is given by

$$\mathrm{ECC} = \mathcal{P}_{\mathrm{same}} \cdot \mathrm{ECC}(1) + \mathcal{P}_{\mathrm{diff}} \cdot \mathrm{ECC}(2) \tag{3}$$

$$\mathrm{ECC} = 0.5 \cdot 100 + 0.5 \cdot 75 = 87.5\%. \tag{4}$$

Hence, the results follows. ■

Based on the above theoretical formulation dealing with all possible three-bit configurations, we can evaluate the probability of occurrence of different CA rules appearing on ith cell to ensure its ECC of single bit noise. Fig. 6 reports the distribution of effective rules in CA rule space for "two attractor basins" GMACA memorizing two patterns with average single bit ECC of 87.5%.

B. Rule Space of GMACA With Multiple Attractor Basins

This subsection analyzes the rule space of GMACA with "multiple attractor basins" formalized in the following theorems. In the subsequent discussions m is denoted as follows: m is the number of distinct neighborhood configurations of the ith $(i = 1, 2, \ldots, n)$ cell; hence, $m \leq 8$.

While the first theorem provides the probability of occurrence of different distinct configurations at ith cell, the remaining theorems establish the error correcting capability of ith cell for different values of m.

Theorem 7: For k number of patterns to be learnt, the probability of occurrence $(\mathrm{PO}(m))$ of m distinct configurations at the ith cell is given by

$$\mathrm{PO}(m) = \frac{\sum_{x=1}^{m-1} {}^4C_x \cdot {}^4C_{m-x}}{\sum_{2}^{y} \sum_{x=1}^{m-1} {}^4C_x \cdot {}^4C_{m-x}} \tag{5}$$

where $y = \min(k, 2^3)$

Proof: Out of total 2^3 configurations, in 4 configurations—$(000, 001, 100, 101)$, state of the ith cell is "0" and in other four configurations—$(010, 011, 110, 111)$, state is "1." So, x number of distinct configurations with state of the ith bit as "0" can be obtained in ${}^4C_x$ ways and rest $(m - x)$ number of distinct configurations with state "1" can be obtained in ${}^4C_{m-x}$ possible ways. So, total possible ways to obtain m distinct configurations is given by $({}^4C_x \cdot {}^4C_{m-x})$ where x varies from 1 to $(m-1)$. For k number of patterns m can vary from 2 to $\min(k, 2^3)$. Hence, the probability of occurrence of m distinct configurations at the ith position is given by

$$\mathrm{PO}(m) = \frac{\sum_{x=1}^{m-1} {}^4C_x \cdot {}^4C_{m-x}}{\sum_{2}^{y} \sum_{x=1}^{m-1} {}^4C_x \cdot {}^4C_{m-x}} \tag{6}$$

where $y = \min(k, 2^3)$. ■

Theorem 8: If the number of distinct configurations of the ith cell is equal to 1, the error correcting capability of the ith cell is 100%.

Proof: Proof follows from that of Theorem 1. ■

Theorem 9: If the number of distinct configurations m of the ith cell of k number of patterns (to be learned) is 2, the error correcting capability is 75%.

Proof: If number of distinct configurations m is equal to 2, the ith cell of k number of patterns behaves as a cell of "two attractor basins." Then, according to Theorem 5, the error correcting capability (ECC(2)) of the ith cell is 75%. ■

Theorem 10: If $m > 2$, the error correcting capability of the ith cell is 66.67%.

Proof: If number of distinct configurations $m > 2$, the ith cell can accommodate noise of $(i-1)$th and $(i-1)$th positions, but cannot recover that of ith position in single time step as there is a collision between state "0" and "1." So, in these cases, the error correcting capability of the ith cell $(\mathrm{ECC}(m > 2))$ is $(2 \cdot k/3 \cdot k)$—that is, 66.67%. Hence, the result follows. ■

Theorem 11: The error correcting capability (ECC) of the ith cell to recover single bit noise in single time step is given by

$$\mathrm{ECC} = \frac{\mathrm{ECC}(1)}{2^{k-1}} + \frac{(2^k - 2)}{2^k} \cdot \sum_{m=2}^{y} \mathrm{PO}(m) \cdot \mathrm{ECC}(m) \tag{7}$$

where k is the number of patterns to be learnt, and $y = \min(k, 2^3)$.

Proof: For k number of patterns, there are total 2^k possible combinations of the ith columns. Out of 2^k combinations, only in two cases, the states of the ith cell of k patterns is same—that is, in case of all zero and all one columns. So, the probability that the state of the ith cell of k patterns is same is given by $\mathcal{P}_{\mathrm{same}} = 1/2^{k-1}$; while that of different is given by $\mathcal{P}_{\mathrm{diff}} = (2^k - 2)/2^k$. The probability of occurrence of m distinct configurations at the ith cell, as per Theorem 7, is equal to $\mathrm{PO}(m)$. If $\mathrm{ECC}(m)$ denotes the error correcting capability of the ith cell with m distinct configurations, then the error correcting capability of the ith cell for k number of patterns is given by

$$\mathrm{ECC} = \mathcal{P}_{\mathrm{same}} \cdot \mathrm{ECC}(1) + \mathcal{P}_{\mathrm{diff}} \cdot \sum_{m=2}^{y} \mathrm{PO}(m) \cdot \mathrm{ECC}(m) \tag{8}$$

$$\mathrm{ECC} = \frac{\mathrm{ECC}(1)}{2^{k-1}} + \frac{(2^k - 2)}{2^k} \cdot \sum_{m=2}^{y} \mathrm{PO}(m) \cdot \mathrm{ECC}(m) \tag{9}$$

where $y = \min(k, 2^3)$. Hence, the result follows. ■

Evolution of *Relation 9*, as shown in Fig. 7 represents the error correcting capability of the ith cell (to recover single bit noise in single time step) for "multiple attractor basins." The graph in Fig. 7 shows that as the value of k (number of patterns to be learned) increases the error correcting capability of the ith cell decreases. Ultimately, it saturates to a critical value $(\mathrm{ECC}_{\mathrm{critical}} \simeq 67\%)$ for $k \geq 8$. Hence, the following theorem.

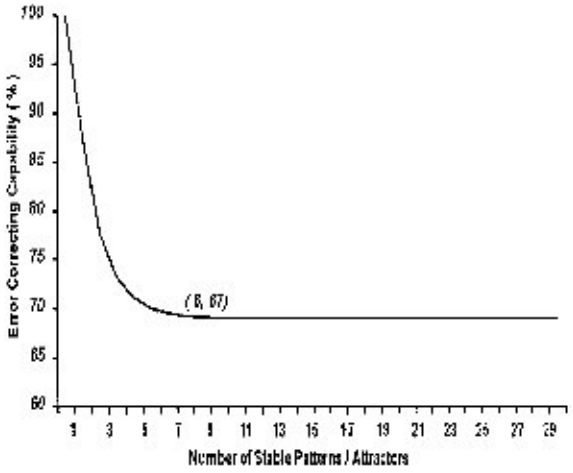


Fig. 7. Error correcting capability of GMACA (theoretically).

Theorem 12: For $k \geq 8$, the error correcting capability of the ith cell is constant.

Proof: According to Theorem 11, the error correcting capability of the ith cell of k number patterns (to be learned) is given by

$$\mathrm{ECC} = \frac{\mathrm{ECC}(1)}{2^{k-1}} + \frac{(2^k - 2)}{2^k} \cdot \sum_{m=2}^{y} \mathrm{PO}(m) \cdot \mathrm{ECC}(m) \tag{10}$$

where $y = \min(k, 2^3)$.

For $k \geq 8$, the term $(1/2^{k-1})$ reduces to zero and the term $((2^k - 2)/2^k)$ tends to one. So, for $k > 8$, the error correcting capability of the ith cell is given by

$$\mathrm{ECC} \simeq \sum_{m=2}^{y} \mathrm{PO}(m) \cdot \mathrm{ECC}(m) \tag{11}$$

where $y = 8$. Equation (11) is independent of the value of k (number of patterns to be learnt) and is equal to 67%. Hence, the result follows. ■

CA Rules: The most effective GMACA rules for "multiple attractor basins" are Rule 76, 140, 196, 200, 204, 205, 206, 220, 236, as well as rules reported in Fig. 6 which are the effective rules for "two attractor basins." So, in CA rule space, the set of most effective rules of "multiple attractor basins" is a subset of

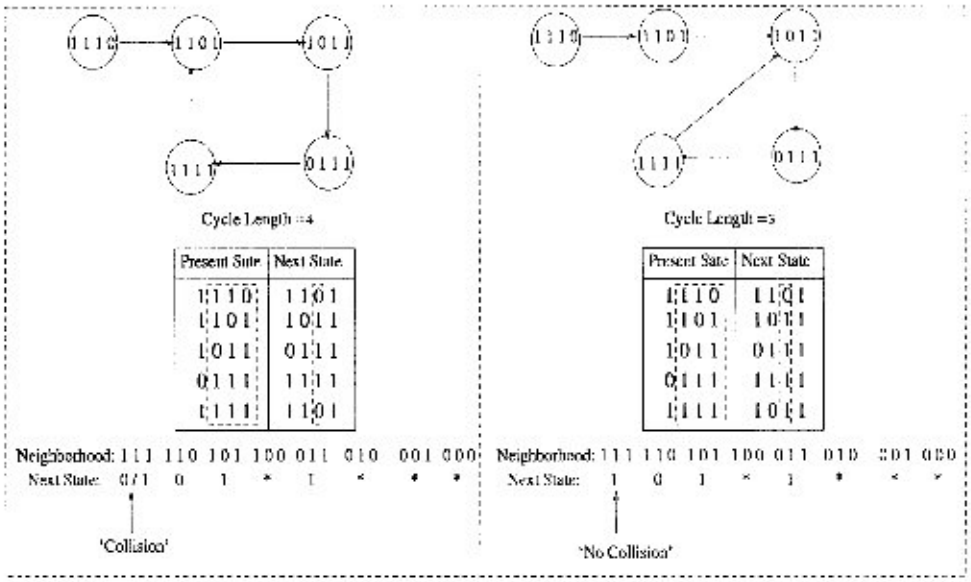


Fig. 8. Example of cycle length reduction of a directed graph (Scheme 1).

"two attractor basins." This fact validates the Assumption (4) of Section IV.

Characterization of rules in CA rule space shows that the collision of state "0" and "1" of a configuration reduces the error correcting capability of the ith cell. If the single bit error of the ith cell is recovered in more than one time step, it may reduce the collision and increase the error correcting capability of the ith cell.

Design of a synthesis algorithm to arrive at such an effective GMACA is a hard problem. So, we fall back on the SA framework to solve this problem. The next section introduces the SA formulation.

V. Simulated Annealing Program for GMACA Evolution

The aim of SA-based evolutionary search is to identify the GMACA that can recognize a given set of patterns without or with specified noise.

SA is a generalization of a Monte Carlo method for examining the equations of state and frozen states of n-body systems [12]. The concept is based on the manner in which liquids freeze or metals recrystallize through the process of annealing.

The GMACA synthesis scheme, as noted below, can be elegantly mapped to this Monte Carlo approach.

1) current state of a thermodynamic system is analogous to the current solution of the synthesis scheme;
2) energy equation for the thermodynamic system is analogous to the "cost function;"
3) finally, the ground state is analogous to the desired GMACA rule space.

So, we map and appropriately tune SA process to generate appropriate graphs out of which desired GMACA can be derived.

In SA an initial temperature $\{\mathrm{Temp}_{\mathrm{initial}}\}$ is set. The temperature decreases exponentially during the process. At each discrete temperature point $\{\mathrm{Temp}_{\mathrm{point}}\}$, k number of graphs are randomly generated, where k is the number of patterns to be learnt. From all these graphs we generate a state transition table and calculate cost function to arrive at the effective rules for different CA cells.

Cost Function: The following cost function is employed to evaluate the quality of solution derived through *Phase I–III* of GMACA synthesis algorithm.

$$C = 1 - \left| \frac{(n_0 \quad n_1)}{(n_0 + n_1)} \right| \tag{12}$$

where n_0 and n_1 are the number of occurrence of state "0" and "1," respectively for a specific present state configuration of a cell. If $n_0 \simeq n_1$, then collision, as defined in Section III.B, is maximum. The value of cost function (C) is maximum and close to 1 for this case; whereas if $n_0 \gg n_1$ or $n_0 \ll n_1$, then C becomes zero and we get GMACA configurations with high degree of precision.

Based on the value of cost function, new sets of graphs are generated. The entire process—generation of graphs, and its evaluation through cost function—that is, *Phase I* to *III*, continues till temperature becomes zero. So, the emphasis is to arrive at graphs with low value of cost function. This demands low collision. Following two schemes are employed to reduce the collision on randomly generated graphs with attractor cycle length $l < l_{\max}$ in *Phase I* of synthesis scheme, where $l_{\max}$ denotes maximum permissible length of the attractor cycle.

Scheme 1: Reduction of Cycle Length of a Given Graph: In this case, we reduce the attractor cycle length of a given graph. Fig. 8 illustrates an example of this technique along with the state transition table and next state function. For this example when the cycle length of the graph is 4, there is a "collision" between state "0" and "1" for "111" configuration of the third cell; whereas when the cycle length is reduced from four to three, there is no "collision" for the same configuration.

Scheme 2: Increment of Cycle Length of a Given Graph: In this case, we increase the cycle length of a given graph. Fig. 9 illustrates an example of this technique. When cycle length of the given graph is 1, there is a "collision" between state "0" and "1" for "111" configuration of the second cell; whereas when

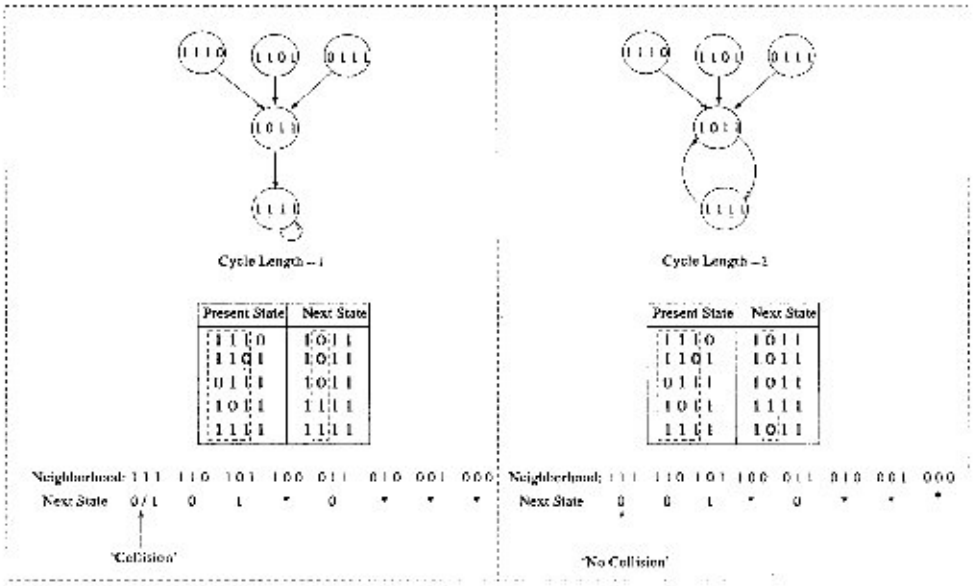


Fig. 9. Example of cycle length increment of a directed graph (Scheme 2).

the cycle length is incremented from 1 to 2, the "collision" disappears.

In *Schemes 1* and *2*, we change the state transition table by changing the cycle length of the given graphs. As a result, the collision between state "0" and "1" of a particular configurations of a cell is changed. Consequently, the cost function is also changed.

The cost value is evaluated according to (12). There are two solutions based on cost value—best solution (BS) and current solution (CS). BS is the solution which has achieved minimum cost value during the lifetime of annealing, while CS is the solution achieved at the present temperature point ($\mathrm{Temp}_{\mathrm{point}}$). A new solution (NS) at the immediate next $\mathrm{Temp}_{\mathrm{point}}$ compares its cost value with that of current solution. If NS has lesser cost value than CS, then NS becomes CS. The new solution (NS) is also compared with BS and if NS is lesser, then NS becomes BS. Even if NS is not as good as CS, NS is accepted with a probability. This step is incorporated to avoid any local minima.

We first derive a rule vector for k number of n-bit patterns (to be learnt) according to theoretical formulations reported in Section IV; and store it as "theoretically derived rule" (TDR). Also, store its cost value as "theoretical cost value" (TCV). Next, we initialize the best solution (BS) as TDR and the cost value of BS as TCV. The evolutionary algorithm of simulated annealing starts after this initialization process that leads to reduced search space of desired GMACA. The complete algorithm is next presented.

Algorithm 1: GMACA Evolution
Input: Pattern Size (n), Attractor Set $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$, Initial temperature ($\mathrm{Temp}_{\mathrm{initial}}$).
Output: Evolved GMACA.
begin:
Step 1. Generate GMACA rule vector according to *Section IV*
Step 2. Store it as "theoretically derived rule" (TDR)
Step 3. Store its cost value as "theoretical cost value" (TCV)
Step 4. Initialize CS as zero rules and BS as TDR.
Step 5. Initialize cost value of CS as zero and cost value of BS as TCV.
Step 6. $\mathrm{Temp}_{\mathrm{point}} = \mathrm{Temp}_{\mathrm{initial}}$
Step 7. while $\mathrm{Temp}_{\mathrm{point}} > 0$
{
(i) if $\mathrm{Temp}_{\mathrm{point}} > 0.50 \times \mathrm{Temp}_{\mathrm{initial}}$
Randomly generate graph as guess solution.
else
Generate a graph by *Scheme 1* or *2*.
(ii) Map a pattern $\mathcal{P}_i$ (to be learnt) on to the cyclic node of a graph.
(iii) Also, map the noisy patterns $\tilde{\mathcal{P}}_i$ to the other nodes of the same graph.
(iv) Repeat (i) to (iii) for k number of graphs, where k is the number of patterns learnt.
(v) Generate state transition table from all the graphs.
(vi) Compute cost value (C) according to (12).
(vii) Generate the GMACA rule.
(viii) NS = GMACA−Rule
(ix) $\delta_{\mathrm{cost}} = \mathrm{cost\ value(NS)} - \mathrm{cost\ value(CS)}$
(x) if $\delta_{\mathrm{cost}} < 0$
{
CS = NS
if cost value(NS) < cost value(BS)
{
BS = NS
}
}
else
CS = NS with prob. $e^{-\delta_{\mathrm{cost}}/\mathrm{Temp}_{\mathrm{point}}}$.
Reduce $\mathrm{Temp}_{\mathrm{point}}$ exponentially.
}

The detailed experimental results and associated performance analysis of next section validate the SA framework we have

TABLE I
COMPUTATION OF MINIMUM VALUE OF MAXIMUM PERMISSIBLE NOISE (r_{max})

Training Noise (r_{max})	Percentage of Recognition					
	For $n = 10$, $k = 4$			For $n = 15$, $k = 4$		
	$r = 1$	$r = 2$	$r = 3$	$r = 1$	$r = 2$	$r = 3$
1	81.36	75.76	44.00	82.97	67.81	45.20
2	79.55	65.27	41.05	81.50	66.46	43.98
3	75.91	61.03	38.20	81.50	66.46	43.46
4	72.46	57.09	28.36	78.13	63.47	41.14
5	69.93	54.11	26.17	74.51	62.03	39.61

Training Noise (r_{max})	Percentage of Recognition					
	For $n = 20$, $k = 5$			For $n = 25$, $k = 6$		
	$r = 1$	$r = 2$	$r = 3$	$r = 1$	$r = 2$	$r = 3$
1	84.57	64.65	44.87	85.91	68.67	51.31
2	83.62	63.95	42.65	79.75	58.88	41.01
3	82.67	61.70	41.55	79.49	58.17	40.58
4	79.31	59.11	37.81	74.13	56.87	37.81
5	73.07	54.82	33.72	71.09	52.77	34.19

Training Noise (r_{max})	Percentage of Recognition					
	For $n = 30$, $k = 7$			For $n = 45$, $k = 10$		
	$r = 1$	$r = 2$	$r = 3$	$r = 1$	$r = 2$	$r = 3$
1	84.88	67.57	50.20	79.30	63.92	51.06
2	81.57	61.99	44.59	75.65	59.41	46.31
3	81.37	61.70	44.56	74.89	59.34	46.01
4	79.65	57.18	41.96	71.15	55.04	42.91
5	74.39	54.19	39.41	69.35	54.89	40.04

set to arrive at the desired solution with feasible computation. While the value of r_{max}—maximum permissible noise at the synthesis phase of GMACA and attractor cycle length of the GMACA give the measure of computation cost of SA evolution, the quality of desired solution can be gauged from the basins of attraction, recognition complexity, etc. of evolved GMACA.

VI. EXPERIMENTAL RESULTS

In this section, we perform extensive experiments based on randomly generated data set for different values of n (number of bits in a pattern) and k (number of patterns to be learnt) to analyze convergence rate of evolutionary algorithm of SA, performance of GMACA based pattern recognizer, and distribution of CA rules for the evolved GMACA. The experiment has been done in Linux (version 2.2.6) environment having machine configuration Pentium II (i686), 400 MHz, 512 KB cache, and 132 MB RAM.

A. Convergence Rate

The time required to arrive at the desired n-cell GMACA increases with the value of r_{max}—the maximum permissible noise. The number of nodes in a graph increases with r_{max} (maximum permissible noise) and so search space for GMACA evolution goes up. Our objective is to identify the optimum value of r_{max} while reducing this search space.

1) Minimum Value of Maximum Permissible Noise: r_{max} specifies the noise level at training/synthesis phase of GMACA. To identify the minimum value of r_{max}, we carry out extensive experiments to evolve pattern recognizable n-cell GMACA for different values of n. For each n, 15 different sets of patterns to be trained are selected randomly. The value of k (number of patterns to be learned) is set in the range of 4 to 10 for different values of n.

TABLE II
EVOLUTION TIME FOR GMACA SYNTHESIS

Size of Pattern (n)	No of Patterns (k)	Initial Temp (T)	Evolution Time (min)
10	4	15	0.43
20	5	15	1.06
30	7	15	1.55
40	10	20	3.01
50	12	25	3.35
60	13	25	4.52
70	15	30	7.03
80	18	35	7.45
90	20	30	9.21
100	23	40	15.68

Table I demonstrate the percentage of convergence/recognition for different noise levels as defined below. Convergence of SA program leads to correct recognition.

Definition 2: Percentage of convergence/recognition at a particular noise of r bit is defined as follows. It is the ratio of number of noisy patterns correctly recognized and total number of noisy patterns with noise of r bits.

Column I of Table I represents noise allowed in training phase, whereas Column II represents the percentage of recognition for different noise value of r bit in identification/recognition phase. The results of Table I clearly establish the following fact: in the training (synthesis) phase if we consider that the patterns are corrupted with only one bit noise, the percentage of convergence at recognition phase is better irrespective of noise level r.

So, the minimum value of r_{max} is set to 1 at synthesis phase for which GMACA based associative memory performs better. So, (1) (Section III-A) reduces to

$$p = 1 + n \tag{13}$$

where n represents number of bits in the patterns to be learnt and p denotes number of nodes in each graph generated in Phase I of Section III-A. This fact validates the Assumption (2) of Section IV. The logic behind this experimental result/assumption follows.

In training phase, if we train patterns with only one bit noise, then total number of patterns in state transition table is $k(1+n)$, where k is the total number of patterns learnt. In that case the collision of state "0" and "1," as explained in Section III–B, is likely to be lesser. This leads to higher probability of generation of best fit GMACA. On the other hand, if we consider noise in more than one bit, then total number entries in the state transition table is higher. This leads to more collisions. As a result the evolutionary algorithm of Simulated Annealing fails to arrive at the GMACA with desired pattern recognition capability.

2) Selection of Attractor Cycle Length: In the context of the observation noted in Section VI-A1, we have analyzed the attractor cycle length of evolved GMACA. We have observed that the length of the attractor cycle of evolved GMACA is equal to one in majority of cases. These results validate the Assumption (1) of Section IV. The result/assumption follows.

If the cycle length of an attractor is equal to one, same neighborhood configurations of the ith cell of an attractor map to same next state more times; in effect collision of state "0" and

TABLE III
ERROR CORRECTING CAPABILITY (ECC) OF EVOLVED GMACA

Size of Pattern (n)	ECC (%)	Number of Iteration
10	92.65	1.001
20	88.91	1.012
30	89.03	1.009
40	93.07	1.013
50	91.13	1.000
60	88.19	1.002
70	86.71	1.011
80	90.84	1.000
90	91.13	1.007
100	87.69	1.001

Size of Pattern (n)	No of Pattern (k)	ECC (%)	Number of Iteration
10	4	81.36	1.201
20	5	84.57	1.071
30	7	84.88	1.034
40	10	78.30	1.030
50	12	78.96	1.001
60	13	80.34	1.026
70	15	77.43	1.015
80	18	76.91	1.018
90	20	78.41	1.010
100	23	75.44	1.004

"1" gets reduced. As a result, the convergence rate of the evolutionary algorithm of Simulated Annealing gets accelerated.

3) Evolution Time: Next, we report the evolution time to arrive at the desired GMACA for different values of n (number of bits) for k number of patterns to be learnt. Table II represents the evolution time to evolve GMACA by SA program. Column I and II of Table II represent the different CA size (n) and number of patterns to be learnt (k), respectively; while Column III depicts the "initial temperature" required to find out the best possible GMACA configuration by SA. In Column IV, we provide the evolution time required to synthesize GMACA for different values of n. The results clearly establish the fact that evolution time of SA program grows within a feasible limit with the increase of n and k.

B. Performance Analysis

This subsection deals with error correcting capability (ECC) of GMACA based associative memory with single bit noise, basins of attraction, and identification/recognition complexity. Extensive experimental results reported here confirm that GMACA can be employed as an associative memory for pattern recognition.

1) Error Correcting Capability (ECC) of Evolved GMACA: Error correcting capability (ECC) of GMACA, as defined in Section IV, is the ratio of number of noisy patterns correctly recognized and ($3 \cdot k$). Whereas the number of iterations required to recognize a noisy pattern is equal to the transient length of the GMACA.

Table III represent the error correcting capability (ECC) of single bit noise ($r = 1$) and the average number of iterations (transient length of the GMACA) required to recognize a noisy pattern corrupted with single bit noise for "two attractor" and "multiple attractor basins," respectively. All the results reported in Table III validate the Assumption (3) of Section IV—that is, the transient length of pattern corrupted with single bit noise is equal to 1.

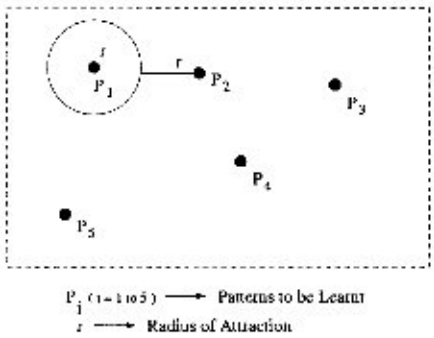


Fig. 10. Example of basins of attraction.

2) Basins of Attraction: The patterns stored in an associative memory must act as attractors. Ideally, they will be the only attractors and will act parsimoniously, so that a given initial state will relax to the nearest attractor.

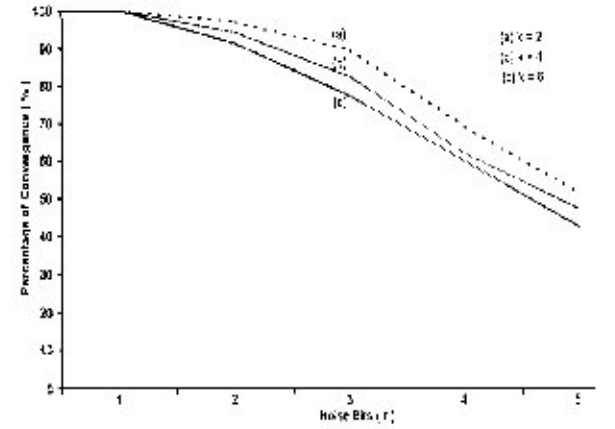
The radius of the basin of attraction of a pattern is defined as the largest hamming distance within which almost all states flow to the pattern. The average of the radii of attraction of each of the stored patterns provides a measure of the pattern recognition capability for a given trained network. The perfect/ideal attractor has basin of attraction is equal to unity, which means that it is possible to move away from any stored pattern, and stay within its basin of attraction up to the point at which another stored pattern becomes nearer.

In Fig. 10, the closest pattern in the training set to P_1 is P_2, at a distance $2r$. Optimal performance occurs when all the vectors within the hypersphere centered on P_1 and radius r, are attempt to P_1. If all patterns stored in a network exhibit this performance, its average basin of attraction is 1.

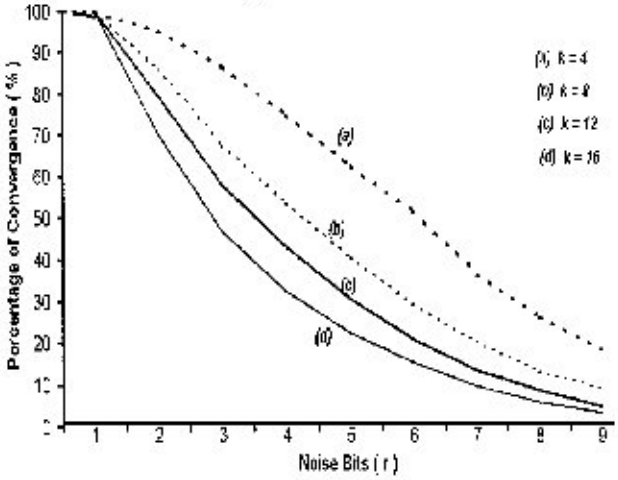
Figs. 11(a)–(c) and 12(a)–(c) represent the performance of recognition/convergence at different noise values in terms of number of bits. Figs. 11(a)–(c) and 12(a)–(c) show the percentage of correct recognition/convergence in the event of noisy patterns input to the GMACA based associative memory. Computation shown is for a fixed hamming distance (HD) between the patterns stored. It is seen that the convergence rate drops about linearly with the amount of noise of the stored patterns.

In Fig. 11(a)–(c), it is shown that, the correct convergence rate also reduces as the number of stored patterns increases for a fixed distortion value of patterns. With the increase in the value of k (the number of patterns to be learned), the performance, as shown in Fig. 11, deteriorates. Fig. 12(a)–(c) represent that, for a fixed value of k, as the number of input bits (n) increases, the performance of recognizing noisy patterns improve. The recognizer performs very well at $n = 100$, but accommodates noise rather poorly at $n = 50$ for both $k = 10$ and 15 [Fig. 12(a)–(c)].

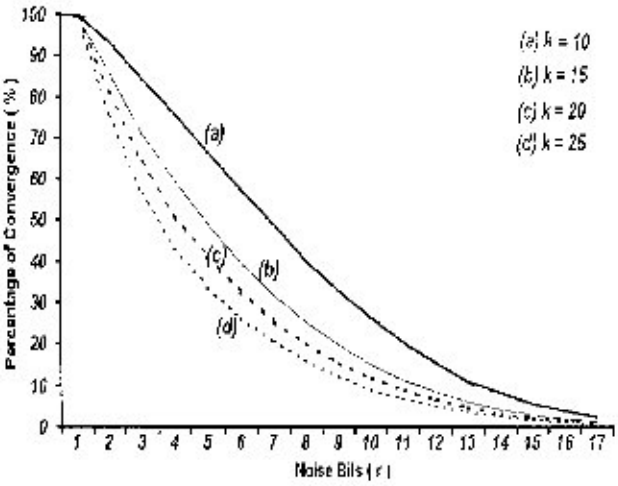
3) Recognition Complexity: In identification process, when a pattern comes as an input (without or with distortion due to noise), the GMACA is run with this pattern for a number of cycles equal to the transient length of the CA and finally settles down to the correct attractor of the incoming pattern. So, the process of identification is independent of the number of patterns learnt (k); it depends on the transient length of given input pattern to the correct attractor.



(a) For $n = 25$

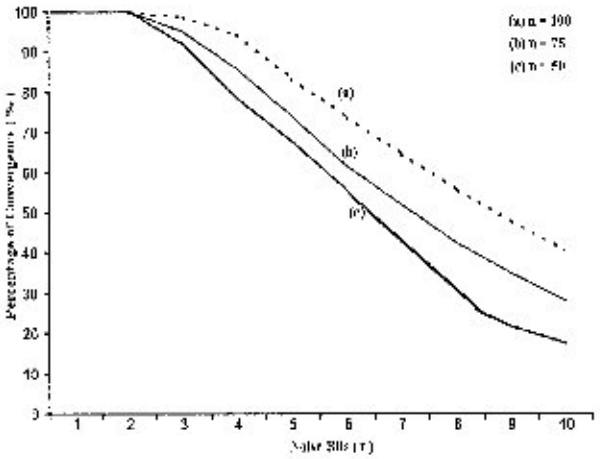


(b) For $n = 50$

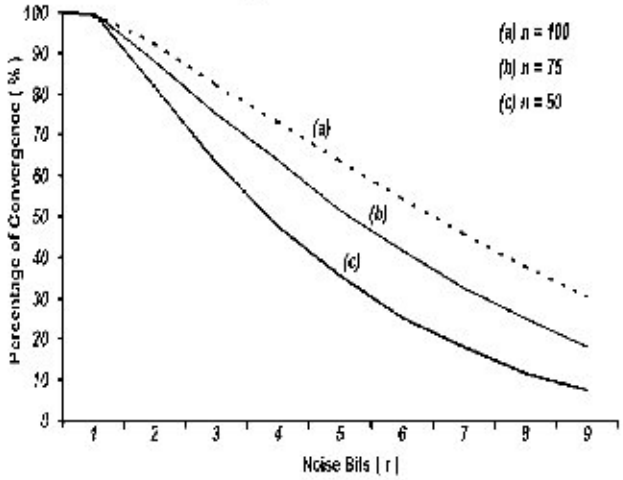


(c) For $n = 100$

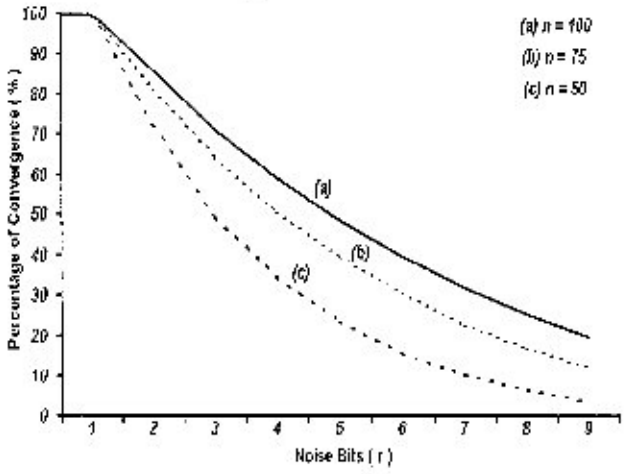
Fig. 11. Graph showing percentage of convergence with noise bits.



(a) For $k = 5$



(b) For $k = 10$

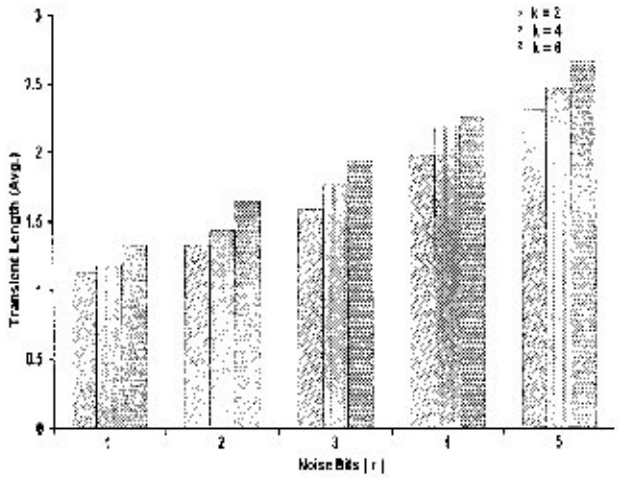


(c) For $k = 15$

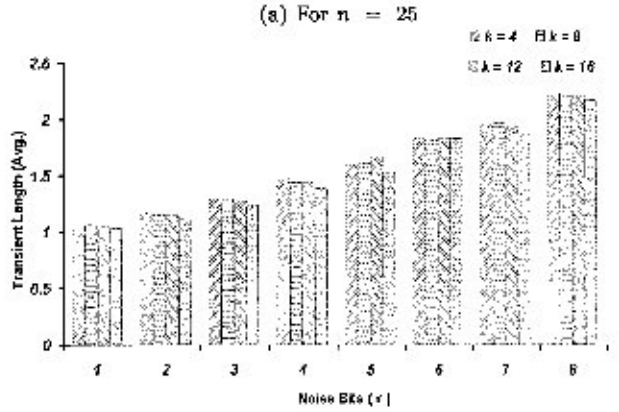
Fig. 12. Graph showing percentage of convergence with noise bits.

Fig. 13(a)–(c) represent the time taken to recognize a noisy pattern for different values of n in terms of the transient length/number of iterations. All the results reported in Fig. 13(a)–(c) ensure that the time taken to recognize a noisy pattern is independent of number of patterns learnt/stored (k). Also, it does not depend on the size of the patterns learnt (n). Only, it depends on the transient length of the CA, which is constant.

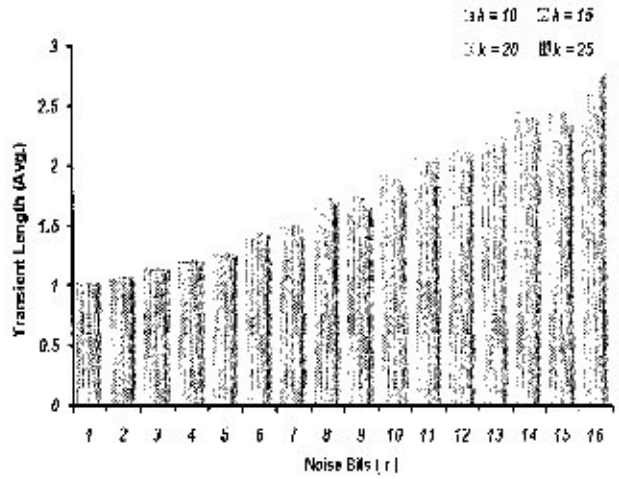
Hence, the cost of computation for entire recognition/identification process is constant.



(a) For $n = 25$



(b) For $n = 50$

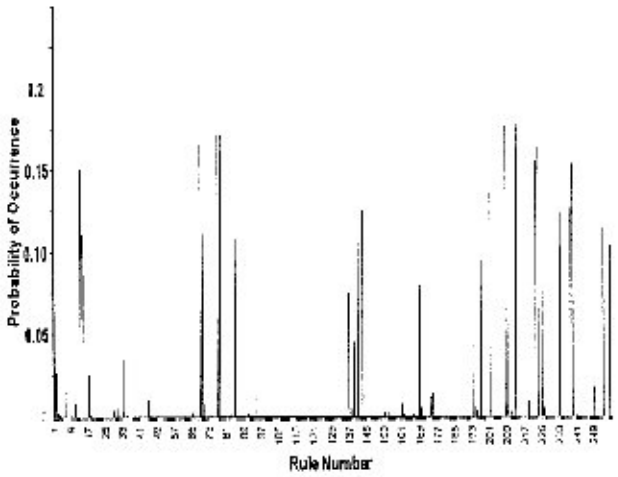


(c) For $n = 100$

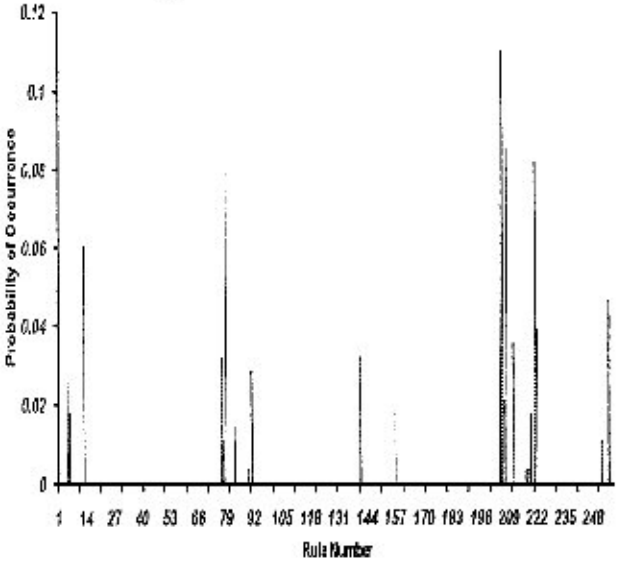
Fig. 13. Graph showing average iteration with noise bits.

C. Distribution of CA Rule of Evolved GMACA

We have studied the rule space of evolved GMACA both for "two attractor" and "multiple attractor basins" in terms of probability of occurrence of different CA rules. The distribution of different CA rules derived from evolutionary algorithm of Simulated Annealing for "two attractor basins" is shown in Fig. 14(a), while Fig. 14(b) shows the distribution for "multiple attractor basins." The distributions reported in Figs. 6 and 14(a)–(b) validate the following fact: the rule space theoretically derived (in Section IV), is a subset of CA rules arrived at through GMACA evolution with SA program.



(a) For two attractor basins



(b) For multiple attractor basins

Fig. 14. Distribution of CA rules in evolved GMACA.

VII. Hardware Architecture of GMACA Based Pattern Recognizer

The hardware architecture of proposed CA-based associative memory model is shown in Fig. 15. The nonlinear CA used for pattern recognition can be realized out of the universal programmable CA (UPCA) structure of Fig. 16. While the basic structure of programmable CA reported in [4] employs only Linear/Additive CA rules with XOR/XNOR logic, an UPCA cell reported in [21] can be configured with any one of the 256 CA rules—Linear/Additive/Non-Linear. It requires one 8:1 MUX. The three-neighborhood configuration of CA cell acts as the control input of this 8:1 MUX.

The rule vector of CA realizing pattern recognizer are stored in program memory as shown in Fig. 15. The input pattern is stored in the input register. The control block configures the UPCA with the rule vector and run for a cycle by taking

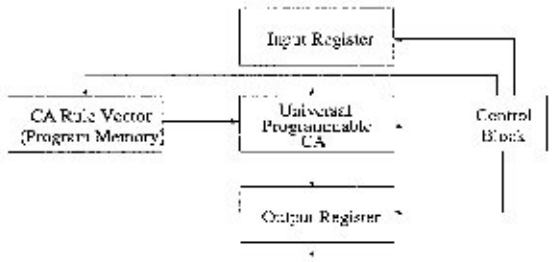


Fig. 15. Hardware architecture of CA-based pattern recognizer.

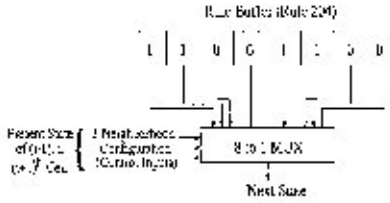


Fig. 16. UPCA cell structure.

the Input Register value. The output of UPCA is stored in the output register. We have written the Verilog code for different blocks and simulated using Cadence Verilog Simulator on Sun Ultra-60 machine. The design has been implemented with 0.25 μm-CMOS technology.

VIII. Conclusion and Future Works

This paper reports the error correcting capability of a CA-based associative memory. Extensive experimental results reported in this paper confirm that CA-based associative memory provides an efficient model for pattern recognition. The simple, regular and modular structure of CA-based associative memory can be employed for high speed online pattern recognition.

Further, the CA-based pattern recognizer can be conveniently employed to address following two problems of practical importance.

1) The proposed CA-based associative memory model can efficiently address the well known "parity" problem. While all the even parity patterns are in one set of attractor basins, another set of attractor basins holds all the odd parity patterns.
2) In word spelling correction problem, the proposed model can be employed considering most commonly used words as the attractors of different attractor basins. Introducing the errors to the extent the model can accommodate, GMACA can be designed for automatic correction of the spelling.

Acknowledgment

The authors wish to acknowledge that is was the probing comments of the referees that refined the work presented in this paper.

References

[1] J. Buhmann, R. Divko, and K. Schulter, "Associative memory with high information content," *Phys. Rev. A*, vol. 39, pp. 2689–2692, 1989.
[2] G. A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Netw.*, vol. 2, no. 4, pp. 243–257, 1989.
[3] M. Chady and R. Poli, "Evolution of Cellular-Automaton-Based Associative Memories,", Tech. Rep. CSRP-97-15, May 1997.
[4] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chatterjee, *Additive Cellular Automata, Theory and Applications*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1997, vol. 1.
[5] N. Ganguly, A. Das, P. Maji, B. K. Sikdar, and P. P. Chaudhuri, "Evolving cellular automata based associative memory for pattern recognition," presented at the Int. Conf. High Performance Computing, Hyderabad, Andhra Pradesh, India, 2001.
[6] N. Ganguly, P. Maji, A. Das, B. K. Sikdar, and P. P. Chaudhuri, "Characterization of nonlinear cellular automata model for pattern recognition," in *Proc. 2002 AFSS Int. Conf. Fuzzy Systems*, Calcutta, West Bengal, India, 2002, pp. 214–220.
[7] N. Ganguly, P. Maji, B. K. Sikdar, and P. P. Chaudhuri, "Generalized multiple attractor cellular automata (GMACA) for associative memory," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 16, pp. 781–795, Nov. 2002.
[8] ——, "Design and characterization of cellular automata based associative memory for pattern recognition," *IEEE Trans. Syst., Man, Cybern. B*, to be published.
[9] J. J. Hopfield, "Neural networks and physical system with emergent collective computational abilities," in *Proc. Nat. Acad. Sci.*, vol. 79, 1982, pp. 2554–2558.
[10] ——, "Pattern recognition computation using action potential timings for stimulus representations," *Nature*, vol. 376, pp. 33–36, 1995.
[11] E. Jen, "Invariant strings and pattern recognizing properties of 1D CA," *J. Stat. Phys.*, vol. 43, 1986.
[12] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[13] W. A. Little and G. L. Shaw, "Analytical study of the memory storage capacity of a neural network," *Math. Biosci.*, vol. 39, pp. 281–290, 1978.
[14] P. Maji, N. Ganguly, A. Das, B. K. Sikdar, and P. P. Chaudhuri, "Study of nonlinear cellular automata for pattern recognition," in *Proc. Cellular Automata Conf.*, Japan, 2001, pp. 187–192.
[15] P. Maji, N. Ganguly, S. Saha, A. K. Roy, and P. P. Chaudhuri, "Cellular automata machine for pattern recognition," in *Proc. 5th Int. Conf. Cellular Automata for Research Industry*, Switzerland, Oct. 2002, pp. 270–281.
[16] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkates, "The capacity of the hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT–33, pp. 461–482, 1987.
[17] J. H. Moore and L. W. Hahn, "Multilocus pattern recognition using one-dimensional cellular automata and parallel genetic algorithms," presented at the Genetic Evolutionary Computation Conf., 2001.
[18] K. Morita and S. Ueno, "Parallel generation and parsing of array languages using reversible cellular automata," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 8, pp. 543–561, 1994.
[19] J. V. Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, Ed, Urbana, IL: Univ. of Illinois Press, 1966.
[20] R. Raghavan, "Cellular automata in pattern recognition," *Inform. Sci.*, vol. 70, pp. 145–177, 1993.
[21] C. Shaw, D. Chatterjee, P. Maji, S. Sen, and P. P. Chaudhuri, "A pipeline architecture for encompression (Encryption+Compression) technology," in Proc. 16th Int. Conf. VLSI Design, India, Jan. 2003, to be published.
[22] G. Weisbuch and F. Fogelman-Soulie, "Scaling laws for the attractors of hopfield networks," *J. Phys. Lett.*, vol. 46, pp. 623–630, 1985.
[23] S. Wolfram, *Theory and Application of Cellular Automata*. Singapore: World Scientific, 1986.

Pradipta Maji received the B.Sc. (Hons.) degree in physics and M.Sc. degree in electronics science from Jadavpur University, India, in 1998 and 2000, respectively.

Currently, he is a Research Scholar in the Department of Computer Science and Technology, Bengal Engineering College (DU), West Bengal, India. His research interests include cellular automata, pattern recognition, data mining, soft computing, artificial intelligence.

Niloy Ganguly received the B.Tech. (Hons.) degree in computer science from IIT, Kharagpur, in 1992 and the M.S. degree in computer science and engineering from Jadavpur University, Calcutta, India, in 1995.

He was Faculty in the Computer Science and Engineering Department in Haldia Institute of Technology, India, from 1997 to 1998, and R. E. College, Durgapur, India, from 1998 to 1999. At present, he is a Lecturer in the M.B.A. Department, Indian Institute of Social Welfare and Business Management, Calcutta, India. His research interests include cellular automata, pattern recognition, data mining, soft computing, artificial intelligence, and digital system design and test.

P. Pal Chaudhuri (SM'84) received the B.E.E. degree in 1963.

He was associated with IBM World Trade Corporation in various capacities until 1975. Subsequently, he switched over to academia and started his career at Indian Institute of Technology (IIT), Kharagpur, India. He took the leading role to initiate the Computer Science and Engineering program at IIT Kharagpur in the late 1970s. During 1986 to 1988, he was the Head/Chairman of the IIT Computer Science and Engineering Department. In 1991 and 1992, he was a Visiting Professor at the University of Illinois, Urbana-Champaign, for one semester and in the subsequent semester, he worked for Cadence Design Systems, India, as a Technical Advisor. During the early 1980s, he initiated a full scale research thrust and established an excellent research base at IIT Kharagpur in the field of VLSI design and testing. Since the late 1980s, he has pioneered the study of the homogeneous structure of cellular automata (CA) and developed matrix algebraic tools in GF(2) to completely characterize the autonomous structure of CA machines. He subsequently applied this theory to develop a wide variety of applications in diverse fields. He has published more than 100 research papers in international journals and conference proceedings. He authored the books *Additive Cellular Automata Theory and Applications Volume 1* (Piscataway, NJ: IEEE Press, 1997) and *Computer Organization and Design* (Englewood Cliffs, NJ: Prentice-Hall), a textbook for undergraduate and graduate level courses. In November 1996, he was invited by Intel Corporation, as a Visiting Faculty. He worked at Intel Research Laboratories, Portland, OR, until the end of 1997. Upon his return, he joined Bengal Engineering College. His research group there has been developing the theory of $GF\{2^n\}$ CA and its applications in a wide variety of fields such as data compression, data security, pattern recognition, simulation of physical systems, VLSI design, and test, etc.