Information Sciences 178 (2008) 2315–2336

Non-uniform cellular automata based associative memory: Evolutionary design and basins of attraction

Pradipta Maji [a,*], P. Pal Chaudhuri [b]

[a] *Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700108, India*
[b] *Cellular Automata Research Laboratory, Techno India Campus, Rajarhat Megacity, Kolkata 700156, India*

Received 20 July 2006; received in revised form 31 December 2007; accepted 3 January 2008

Abstract

This paper presents the synthesis and analysis of a special class of non-uniform cellular automata (CAs) based associative memory, termed as generalized multiple attractor CAs (GMACAs). A reverse engineering technique is presented for synthesis of the GMACAs. The desired CAs are evolved through an efficient formulation of genetic algorithm coupled with the reverse engineering technique. This has resulted in significant reduction of the search space of the desired GMACAs. Characterization of the basins of attraction of the proposed model establishes the sparse network of GMACAs as a powerful pattern recognizer for memorizing unbiased patterns. Theoretical analysis also provides an estimate of the noise accommodating capability of the proposed GMACA based associative memory. An in-depth analysis of the GMACA rule space establishes the fact that more heterogeneous CA rules are capable of executing complex computation like pattern recognition.

Keywords: Cellular automata; Genetic algorithm; Pattern recognition; Associative memory; Basins of attraction

1. Introduction

Cellular automata (CAs) provide new means for doing computation more efficiently – in terms of speed, cost, power dissipation, information storage, and solution quality [33]. They have been proposed to study the general phenomenological aspects, including communication, computation, construction, growth, reproduction, competition, and evolution [14,15,24,30]. CAs also provide an excellent tool for modeling physical phenomena by reducing them to their basic, elemental laws [21,29,32].

Interesting computational properties of the CA model have inspired us to investigate new research area such as pattern recognition. The associative memory model provides a solution to the problem of pattern recognition, where time to recognize a pattern is independent of the number of patterns learnt (stored) [1]. This model divides the entire state space into some pivotal points that represent the patterns learnt. The pivotal

[*] Corresponding author. Tel.: +91 33 2575 2044; fax: +91 33 2578 8699.
E-mail addresses: pmaji@isical.ac.in (P. Maji), palchau@carltig.res.in (P. Pal Chaudhuri).

points can be viewed as stable states or attractors. The states close to a pivotal point get associated with the corresponding learnt pattern and are referred to as transient states. Identification of an input pattern amounts to traversing the transient path 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 becomes independent of the number of patterns learnt. A logical evolution is to build the associative memory model for pattern recognition.

The design of CA based associative memory has so far mostly concentrated around uniform CAs [2,13,18] with same rule applied to each of the CA cells. The structure of uniform rule restricts the CAs to evolve as a general purpose pattern recognizer although they have been able to memorize some specific patterns [2,18]. Some works on non-uniform CAs have also been reported in [26,27]. However, none of the works reported the evolution of CAs as a general purpose pattern recognizer [2,18,26,27]. The design of CA based model for pattern recognition is reported in [3,19,20,22,25]. The search for associative memory model for pattern recognition around simple structure of CAs has been reported in [9,10,17]. The results reported in [9,10,17] have explored the potential application of 3-neighborhood non-uniform CAs, referred to as generalized multiple attractor CAs (GMACAs), as an efficient pattern recognizer. In [17], the error correcting capability of the GMACAs at single bit noise has been reported considering the attractor cycle of the GMACAs with single or multiple nodes. Also, the simulated annealing program has been used to synthesize the desired GMACA based associative memory [17]. However, these preliminary works have not addressed the following aspects:

(i) exponential growth of the search space of the desired CAs;
(ii) theoretical analysis of the basins of attraction of the GMACAs; and
(iii) theoretical analysis of the desired GMACA rule space.

One of the major impediments of the CA based computing model stems from the difficulty of utilizing its complex behavior to perform useful computations. The search for appropriate CAs displaying a specific behavior or performing a particular computation is a challenging task. This problem has severely restricted CA applications for real life problems. Automated search for the desired CAs performing a specific task would greatly enhance the viability of CA applications. Thus, one of the major challenges in the CA research is to develop a synthesis methodology to find out acceptable solution that will enable researchers to explore new application areas. The genetic algorithm (GA) [11,12] provides a useful framework to arrive at the desired CA rule structures appropriate to model a physical system. The GA has been successfully applied to search for the desired CAs performing a specific task such as density classification [4,6,18,26], sequence generation [28], cryptography [31], modeling recrystallization process [7,23], pattern recognition [9,10], and so on.

In this paper, we develop the GMACA based associative memory model around single cycle attractor. An efficient formulation of the GA is coupled with a reverse engineering technique to synthesize the desired GMACA configurations. The basins of attraction of the GMACAs, that is, the error correcting capability at multiple bit noise, is also analyzed. Theoretical formulation of the GMACA rule space establishes the fact that the CAs with more heterogeneous rules are capable of executing complex computation like pattern recognition.

The structure of the rest of this paper is as follows. Section 2 briefly introduces necessary notions of CA, GMACA, and GMACA based associative memory model. In Section 3, a reverse engineering technique is reported for synthesis of the desired GMACAs. A GA based evolutionary synthesis scheme is next presented in Section 4 employing the proposed reverse engineering technique. Section 5 reports the theoretical analysis of the basins of attraction of the GMACA based associative memory and the associated rule space, while a few case studies and a comparison with [9,10] are presented in Section 6. Finally, concluding remarks are given in Section 7.

2. Cellular automata

A cellular automaton (CA) [33] consists of a number of cells organized in the form of a lattice. It evolves in discrete space and time. The next state of a cell depends on its own state and the states of its neighboring cells. In a 2-state 3-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 conventionally called the rule number for the cell [33]. Out of 256 rules, two rules 120 and 232 are illustrated below:

Neighborhood :	111	110	101	100	011	010	001	000	Rule
(i) Next State :	0	1	1	1	1	0	0	0	120
(ii) Next State :	1	1	1	0	1	0	0	0	232

The first row lists the possible 2^3, that is, 8 combinations of the present states (left, self, and right) for a 3-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. An n-cell CA is configured with the rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$, where the ith cell is configured with the rule $\mathscr{R}_i$, each $\mathscr{R}_i$ being one of the possible 256 rules. The complete set of rules is referred to as the CA rule space. A few fundamental terminologies are introduced next that are referred to different sections of this paper:

- If all the CA cells obey the same rule, then the CA is said to be a uniform CA; otherwise it is a non-uniform (hybrid) CA.
- A CA is said to be a null boundary CA, if the left (right) neighbor of the leftmost (rightmost) cell is connected to logic 0-state.
- A CA is said to be a periodic boundary CA, if the left (right) neighbor of the leftmost (rightmost) cell is the rightmost (leftmost) cell.

In this paper, we use null boundary, non-uniform (hybrid) CA based associative memory for pattern recognition.

2.1. Generalized multiple attractor CA

A GMACA is a non-uniform (hybrid) CA (different rules applied to different cells) and can efficiently model an associative memory [9,10,17] to perform pattern recognition task. Fig. 1 illustrates the state space of a 4-cell GMACA with the rule vector $\langle 202, 168, 218, 42 \rangle$, that is, the rule 202 is applied on leftmost cell, followed by the rule 168 on next one, and so on. The state space of this CA is divided into two attractor basins, basin-1 and basin-2, built around attractor-1 and attractor-2. A GMACA has attractors of cycle length greater than or equal to 1. For the GMACA of Fig. 1, 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.

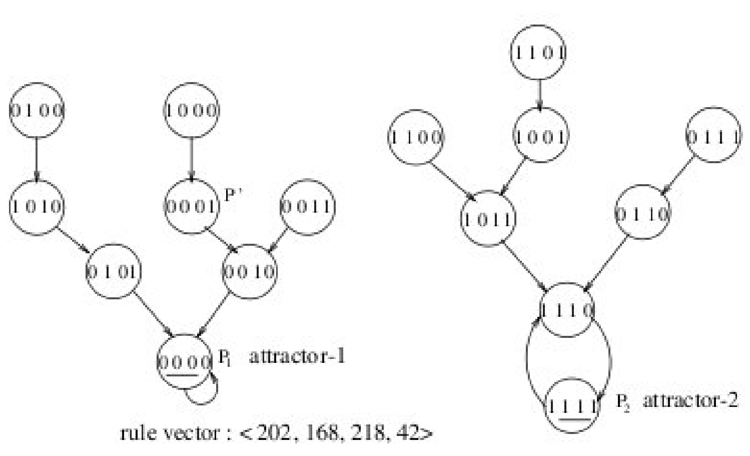


Fig. 1. State space of a GMACA divided into 2-attractor basins.

2.2. GMACA modeling an associative memory

A GMACA having its state space distributed into disjoint basins (Fig. 1) with transient and attractor states, models an associative memory [9,10,17]. The patterns to be learnt, are modeled as pivotal points/stable states. 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 $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$ is equivalent to design of a GMACA with the pivotal points as the states in different attractor cycles. Any transient point $\acute{\mathscr{P}}_x$, with limited distance from a pivotal point $\mathscr{P}_x$, can be considered as a pattern with some noise. The correct output $\mathscr{P}_x$ can be produced in time proportionate to the traversal of the GMACA from the state $\acute{\mathscr{P}}_x$. In the present work we assume hamming distance (HD) as the similarity metric. A GMACA satisfying the following two relations models an associative memory storing the set $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$:

- **R1:** Each attractor basin of the GMACAs should contain one and only one pattern $\mathscr{P}_x$ to be learnt in its attractor cycle; the corresponding basin is referred to as the $\mathscr{P}_x$-basin.
- **R2:** The HD of each state $\acute{\mathscr{P}}_x \in \mathscr{P}_x$-basin with $\mathscr{P}_x$ is lesser than the HD of $\acute{\mathscr{P}}_x$ with any other $\mathscr{P}$'s, that is,

$$\mathrm{HD}(\acute{\mathscr{P}}_x, \mathscr{P}_x) < \mathrm{HD}(\acute{\mathscr{P}}_x, \mathscr{P}_y), \quad \mathscr{P}_y \in \mathscr{P} \text{ and } \mathscr{P}_y \neq \mathscr{P}_x. \tag{1}$$

While the relation **R1** ensures uniqueness of the stored patterns in an attractor cycle, the relation **R2** ensures recognition of patterns with distortion due to noise.

Example 1. The example GMACA of Fig. 1 is used to illustrate the above two relations. To recognize two patterns $\mathscr{P}_1 = 0000$ and $\mathscr{P}_2 = 1111$ of length 4, we first synthesize a CA rule vector $\mathscr{R} = \langle \mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3, \mathscr{R}_4 \rangle$ for which the state transition behavior of the GMACA is similar to that of Fig. 1, that maintains both **R1** and **R2**. It learns two patterns, $\mathscr{P}_1 = 0000$ and $\mathscr{P}_2 = 1111$. The state $\acute{\mathscr{P}} = 0001$ has the HD 1 and 3 with $\mathscr{P}_1$ and $\mathscr{P}_2$, respectively. Let $\acute{\mathscr{P}}$ 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. 1 is loaded with $\acute{\mathscr{P}} = 0001$. The GMACA returns the desired pattern $\mathscr{P}_1$ after two time steps.

2.3. Existing method to synthesize GMACAs

In [9,10], a GA based searching scheme has been employed to evolve the GMACAs tuned for a set of patterns to be recognized. The search is made to arrive at a rule vector of the GMACA that satisfies both the relations **R1** and **R2**. All possible combinations of the non-uniform (hybrid) CA rules (256^n) are the candidates for initial population. Since the search space is exponentially large, the GA has been employed and appropriately tuned to arrive at the desired GMACAs.

The GA starts with an initial population of 50 chromosomes, each chromosome being a CA rule vector [9,10]. So, the length of each chromosome is n, n being the number of CA cells. Each cell of the CA can be configured with any one of the 256 rules. Fig. 2 represents an example of a chromosome format for the GA formulation employed in [9,10] with $\mathscr{R}_i$ as the CA rule applied on the ith cell. The initial population of the GA has been generated based on the CA rule configuration [10]. The fitness of a particular chromosome in a population is determined by the HD between the attractor $\widetilde{\mathscr{P}}_x$, evolved for a state from the run (generation), and the desired attractor $\mathscr{P}_x$. The chromosome is run with 300 randomly chosen initial configurations and the fitness of the CA is determined by averaging the fitness for each individual initial configuration. Each chromosome evolves over several generations under selection, crossover (single point), and mutation (single

60	100	175	90	232	212	192
R_1	R_2	R_3	R_4	R_5	R_6	R_7

Fig. 2. An example chromosome format where $\mathscr{R}_i$ represents the CA rule applied on the ith cell.

point). The selection is done by the elitist model proposed by De Jong [5] to preserve the best chromosome. If the best chromosome of the current generation is not present in the next generation, it is included as the new member of the next generation. In effect, the overall performance is improved significantly.

3. Synthesis of GMACAs using reverse engineering technique

The synthesis algorithm of the GMACAs can be viewed as the training phase of the CA based pattern recognizer to recognize a given pattern set. The synthesis of the GMACAs is presented using a reverse engineering technique supported with the GA evolution.

3.1. Reverse engineering technique

The algorithm accepts the pattern set $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$ to be learnt as the input, each $\mathscr{P}_x$ $(x = 1, 2, \ldots, k)$ being an n-bit pattern. The output is the GMACA rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$ specifying the rule number applied for each of the cells. The following three phases illustrate the synthesis scheme.

Phase I. The state transition diagram of a GMACA can be conceived as a graph consisting of multiple sub-graphs (Fig. 1). Each node of a sub-graph is represented as a state of n-bit string. The direction of edges connecting the nodes flow inward from the transient states to the attractor, and then clockwise around the attractor cycle. So, an attractor basin can be portrayed as a sub-graph.

In Phase I, we randomly generate k $(1 < k < 2^n)$ number of sub-graphs, each representing an attractor basin of the candidate GMACA to be synthesized. The cycle length of each sub-graph is considered as 1. That is, we consider single cycle (point) attractor, where the space–time pattern has completely stabilized. Analogous to a GMACA basin (marked as the $\mathscr{P}_x$-basin) a sub-graph g_x has a unique pattern $\mathscr{P}_x$ in its cyclic node, while the noisy patterns of $\mathscr{P}_x$ are considered as the transient states of the $\mathscr{P}_x$-basin. The number of nodes p of each sub-graph is equal to the number of states in the attractor basin, that is, the patterns without or with specified noise. So

$$p = \sum_{r=0}^{r_{\max}} \binom{n}{r}, \tag{2}$$

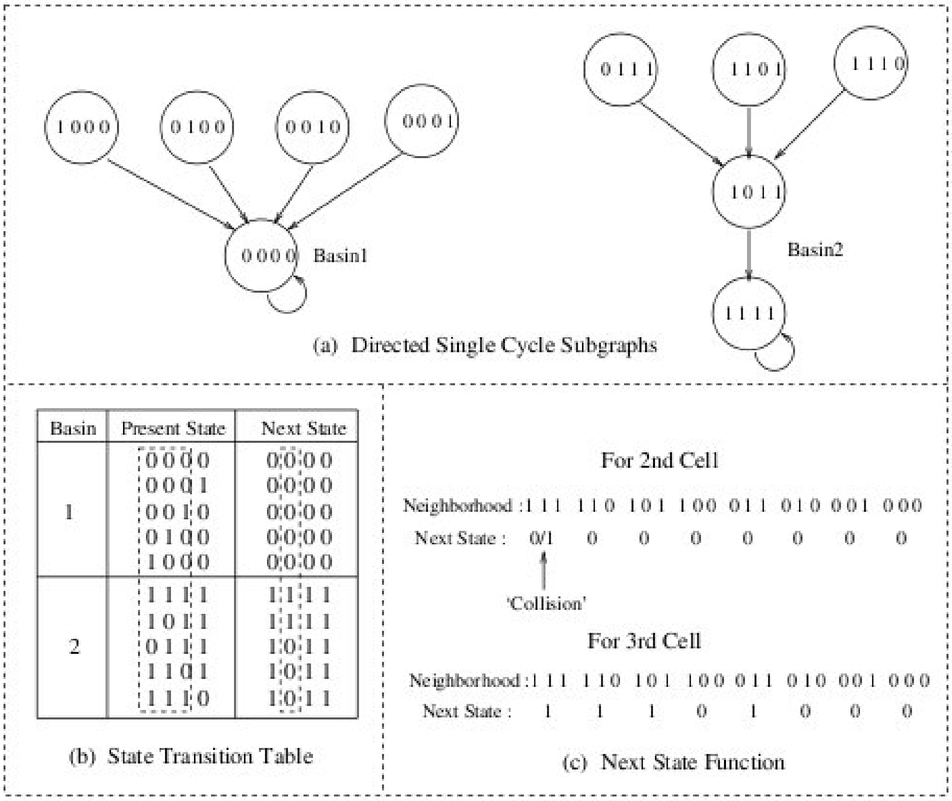
where r is the number of noisy bits and $r_{\max}$ $(0 \leqslant r_{\max} \leqslant n)$ is the maximum permissible noise, that is, a noisy pattern say $\acute{\mathscr{P}}_x$ (where HD $(\acute{\mathscr{P}}_x, \mathscr{P}_x) = r_{\max}$) should be correctly recognized as $\mathscr{P}_x$ by the GMACA based model. In other words, $\acute{\mathscr{P}}_x$ must be a transient state of the $\mathscr{P}_x$-basin. As we consider single cycle attractor, the total number of possible distinct sub-graphs G with p number of nodes is given by [16]

$$G \simeq \sum_{h=1}^{h_{\max}} \binom{p-2}{h-1}, \tag{3}$$

where h is the height of the sub-graph and $h_{\max}$ $(1 \leqslant h_{\max} \leqslant p-1)$ is the maximum permissible height.

After mapping a pattern $\mathscr{P}_x$ $(\mathscr{P}_x \in \mathscr{P})$ to the cyclic node of a sub-graph g_x, we randomly map noisy patterns of $\mathscr{P}_x$ (transient states of the $\mathscr{P}_x$-basin) to different nodes of the same sub-graph. Note that the $\mathscr{P}_x$-basin covers the transient states with permissible noise of r bits ($r = 0$ to $r_{\max}$) added to $\mathscr{P}_x$. If a noisy pattern is equally apart from more than one patterns to be learnt, then the desired attractor basin of the noisy pattern should be selected randomly. The following example illustrates the underlying principle of Phase I of the synthesis scheme.

Example 2. Consider two pattern to be learnt $\mathscr{P}_1 = 0000$ and $\mathscr{P}_2 = 1111$ of length 4. That is, $n = 4$ and $k = 2$. If we set the maximum permissible noise $r_{\max} = 1$, then the number of nodes p of each sub-graph will be 5 and the height h of each sub-graph may vary from 1 to 4. Hence, the total possible distinct sub-graphs $G \simeq 8$. Each attractor basin of the GMACA, which can be considered as a sub-graph, may be any one of the eight possible sub-graphs. Out of eight possible sub-graphs, two sub-graphs with heights 1 and 2 are presented in Fig. 3(a) as an example. The patterns to be learnt $\mathscr{P}_1$ and $\mathscr{P}_2$ are mapped to the cyclic nodes of two sub-graphs considering them as the attractors of two attractor basins. The noisy patterns corresponding to $\mathscr{P}_1$ and $\mathscr{P}_2$, 0001, 0010, 0100, 1000 and 1110, 1101, 1011, 0111, respectively, are considered as the transient states of the $\mathscr{P}_1$ and $\mathscr{P}_2$-basins. These are mapped to two attractor basins as shown in Fig. 3(a).



Basin	Present State	Next State
1	0 0 0 0	0 0 0 0
	0 0 0 1	0 0 0 0
	0 0 1 0	0 0 0 0
	0 1 0 0	0 0 0 0
	1 0 0 0	0 0 0 0
2	1 1 1 1	1 1 1 1
	1 0 1 1	1 1 1 1
	0 1 1 1	1 0 1 1
	1 1 0 1	1 0 1 1
	1 1 1 0	1 0 1 1

(b) State Transition Table

For 2nd Cell

Neighborhood :	1 1 1	1 1 0	1 0 1	1 0 0	0 1 1	0 1 0	0 0 1	0 0 0
Next State :	0/1	0	0	0	0	0	0	0

'Collision'

For 3rd Cell

Neighborhood :	1 1 1	1 1 0	1 0 1	1 0 0	0 1 1	0 1 0	0 0 1	0 0 0
Next State :	1	1	1	0	1	0	0	0

(c) Next State Function

Fig. 3. Randomly generated graph with state transition table and rules for 4-bit patterns.

Phase II. From the k sub-graphs (state transition graph) generated in Phase I, a state transition table is derived. Fig. 3(b) represents a state transition table which is derived from the two sub-graphs of Fig. 3(a). If the number of nodes of a sub-graph is p, then the total number of entries in the state transition table is $(k \cdot p)$. For example, as in case of Fig. 3(a), the number of attractors $k = 2$ and the number of states in each attractor basin $p = 5$, the total number entries in the state transition table of Fig. 3(b) is 10.

Phase III. Generate the rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$ of the GMACA cells from the state transition table. Consider the ith cell whose rule $\mathscr{R}_i$ is to be identified. As we consider 3-neighborhood dependency, to identify the rule $\mathscr{R}_i$, we concentrate only on 3 columns, $(i-1)$th, ith and $(i+1)$th of all the $(k \cdot p)$ entries of the state transition table. 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.

Example 3. Fig. 3(c) represents the neighborhood configurations along with the next state of second and third cells of the patterns (states) of two attractor basins. For the third cell, there is no collision between state 0 and 1 for any 2^3 configurations; whereas for 111 configuration, the next state of the second cell is 0 in one case and 1 in the next case, that is, $n_0 = 1$ and $n_1 = 1$. So, for 111 configuration, the next state may be 0 or 1 generating an instance of collision.

3.2. Resolution of collision

In order to resolve the conflict, the following heuristics is introduced. 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. If $n_0 \gg n_1$ or $n_0 \ll n_1$, the collision between them is minimum. In that case, the next state of a cell is 0 if $n_0 > n_1$, and 1 otherwise. The algorithm for synthesis of the GMACAs through the reverse engineering technique follows next.

Algorithm 1

```
GMACA_Synthesis
– Input: Pattern size ($n$), pattern set $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$.
– Output: Value of $n_0$ and $n_1$ for the $i$th cell ($i = 1, 2, \ldots, n$), rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$.
    (i) Generate a directed single cycle sub-graph $g_x$ with $p$ number of nodes (Eq. (2)).
   (ii) Map a pattern $\mathscr{P}_x$ (attractor) to the cyclic node of $g_x$.
  (iii) Also, map noisy patterns $\acute{\mathscr{P}}_x$ (transients) of $\mathscr{P}_x$ to the other nodes of $g_x$.
   (iv) Repeat Step (i) to (iii) for $k$ number of attractors.
    (v) Generate state transition table from all the sub-graphs.
   (vi) Calculate $n_0$ and $n_1$ for each $i$th ($i = 1,2,\ldots,n$) cell.
  (vii) Generate rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$ while resolving collision instances.
 (viii) Stop.
```

The resolution of collision gives a measure of the efficiency of the reverse engineering technique. The efficiency can be enhanced through appropriate mapping of noisy patterns to the graph. All possible combinations of the sub-graphs for k number of patterns to be learnt (G^k) are the candidate solutions. As the search space is exponentially large, we fall back on the evolutionary algorithm of the GA [11,12] to address this problem. The GA is employed to evolve the appropriate mapping function.

4. Genetic algorithm for GMACA evolution

Genetic algorithm (GA) [11,12] is a stochastic search and optimization technique designed as per the mechanics of natural selection and genetics. In order to apply a GA effectively, two types of decisions must be made. The first is the choice of representation – solutions to the problem must be represented as strings and genetic search operators (e.g. crossover, mutation, selection) must be chosen appropriately. The solutions of the search space, encoded in string like structures, are referred to as chromosomes. The chromosomes undergo evolution for a number of generations. The reproductive plan essentially consists of tuning the three genetic operators for a specific application domain. In order to employ this evolution methodology for the GMACAs synthesis, the essential prerequisite is to encode the solution (GMACAs) in a higher level of representation, defined by a series of sub-graphs, each one representing a gene in the chromosome. The encoding scheme is illustrated next.

4.1. Encoding scheme

Let the input be k number of n-bit patterns to be learnt $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$. The output, as formulated earlier, is an n-cell GMACA with k number of attractor basins. In a population, a chromosome consists of k number of genes, each gene is a single cycle sub-graph with p number of nodes, where p is given by Eq. (2). That is, the number of nodes of each sub-graph is a function of n (number of CA cells). Hence, in the current GA formulation the chromosome depends on both the parameters n and k, while the chromosome for the GA employed in [9,10] depends only on n and is independent of the value of k.

Fig. 4 illustrates the encoding scheme. Each gene g_x of the chromosome represents a basin of a pattern $\mathscr{P}_x$ to be learnt and consequently mapped to an attractor. Therefore, each gene g_x besides keeping the information of the attractor $\mathscr{P}_x$ also stores that of other noisy patterns (transient states) to be covered by the $\mathscr{P}_x$-basin at

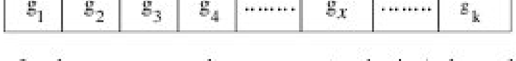


Fig. 4. An example of a chromosome, each g_x represents a basin (sub-graph) of an attractor $\mathscr{P}_x$.

different nodes of the same sub-graph. The fitness of any graph is evaluated with the help of the following fitness function employed for the GA evolution.

4.2. Fitness function and selection

The fitness of any GMACA rule vector synthesized through Phase I,II.III is evaluated through the fitness function F as defined below:

$$F = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(n_{0i} - n_{1i})}{(n_{0i} + n_{1i})}\right|, \tag{4}$$

where n represents the number of CA cells, n_{0i} and n_{1i} are the occurrence of state 0 and 1 for a particular configuration of the ith cell, respectively. If $n_{0i} \simeq n_{1i}\ \forall i$, the value of F is close to zero; whereas if $n_{0i} \gg n_{1i}$ or $n_{0i} \ll n_{1i}\ \forall i$, the value of F becomes 1 and we get GMACA configurations with high degree of precision.

Selection is done by the standard ranking approach, which is a proportionate method like roulette wheel. The probabilities are calculated on the basis of ranking of the individuals in terms of the fitness function, instead of the fitness function itself. Elitism is incorporated in the selection process to prevent oscillation of the fitness function with generation. The fitness of the best individual of a new generation is compared with that of the current generation. If the later has a higher value the corresponding individual replaces a randomly selected individual in the new population.

4.3. Crossover

The crossover implemented is the conventional single point technique normally used in evolutionary algorithms, as Fig. 5 illustrates. The algorithm takes two chromosomes from the present population (PP) and forms the resultant chromosomes, by setting a crossover point randomly. The sequence of sub-graphs from beginning of chromosome to the crossover point is copied from one parent while the rest is copied from the second parent resulting in two offsprings (new chromosomes) for the next population (NP).

4.4. Mutation

The mutation algorithm emulates the normal mutation scheme. It makes some minimal change in the existing chromosome of the PP to form a new chromosome for the NP. Similar to conventional single point mutation, the chromosome in the current GA formulation is mutated at a single point. In this case, one sub-graph of a chromosome is randomly selected and replaced by a new sub-graph producing a new chromosome as a

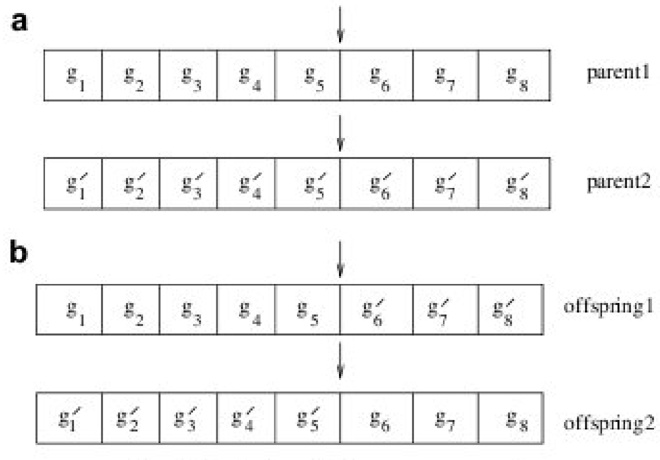

a
g1 g2 g3 g4 g5 g6 g7 g8
parent1
g1' g2' g3' g4' g5' g6' g7' g8'
parent2
b
g1 g2 g3 g4 g5 g6' g7' g8'
offspring1
g1' g2' g3' g4' g5' g6 g7 g8
offspring2

Fig. 5. Illustration of the crossover operator.

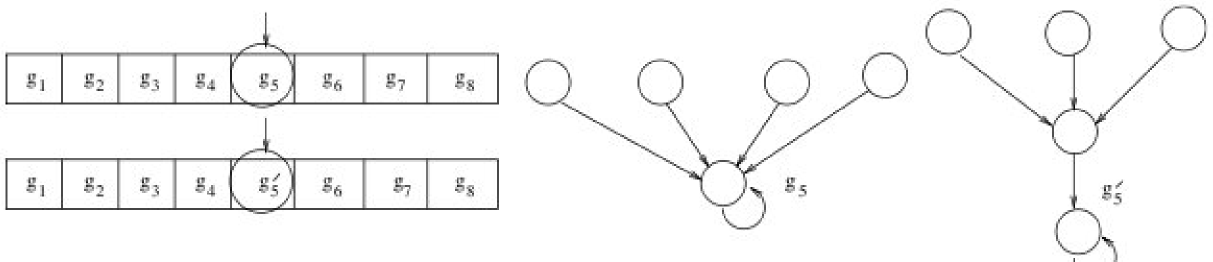


Fig. 6. Example of the mutation operator.

solution. The new sub-graph consists of same number of nodes, as that of original sub-graph, with different edges. Fig. 6 represents an example of mutation on a randomly selected sub-graph (g_5) of a chromosome along with the new sub-graph $(\acute{g}_5)$.

In the proposed method, the GA parameters used are as follows:

Maximum generation: 1000; Population size: 50;
Chromosome length: k, where k is the number of patterns to be learnt;
Number of elite chromosomes: 5; Probability of crossover: 0.8; Probability of mutation: 0.001.

The parameters are held constant across all runs. Unbiased initial population is generated randomly spreading over entire variable space in consideration. The complete algorithm for evolving GMACAs through the GA follows next.

Algorithm 2

```
Evolving_GMACA
– Input: Pattern size $(n)$, pattern set $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$, Maximum Generation $(G_{\max})$.
– Output: GMACA rule vector $\mathscr{R} = \langle \mathscr{R}_1, \ldots, \mathscr{R}_i, \ldots, \mathscr{R}_n \rangle$.
    (i) Generate 50 new chromosomes for initial population (IP).
    (ii) Initialize generation counter $GC = 0; PP \leftarrow IP$.
    (iii) Calculate $n_0$ and $n_1$ for each cell $i$ $(i = 1, 2, \ldots, n)$ according to Algorithm 1 and generate rule vector.
    (iv) Compute fitness value $F$ according to Eq. (4).
    (v) Repeat Steps (iii) and (iv) for each chromosome of $PP$.
    (vi) Rank chromosomes in order of fitness.
    (vii) Store rule and corresponding generation number for which the value of $F = 1$.
    (viii) If $F = 1$ for at least one chromosome, then go to Step (xiv).
    (ix) Increment generation counter $(GC)$.
    (x) If $GC > G_{\max}$ then go to Step (xiii).
    (xi) Form NP by selection, crossover, and mutation.
    (xii) $PP \leftarrow NP$; Go to Step (iii).
    (xiii) Store rule and corresponding generation number for which $F$ is maximum.
    (xiv) Stop.
```

4.5. Convergence rate

The time required to arrive at the desired n-cell GMACAs with k-attractor basins increases with the values of the maximum permissible noise $r_{\max}$ and the maximum permissible height of each sub-graph $h_{\max}$. While the number of nodes in a sub-graph increases with the value of $r_{\max}$, the number of possible sub-graphs increases with $h_{\max}$. So, the search space for the GMACA evolution goes up. The objective here is to identify the optimum values of $r_{\max}$ and $h_{\max}$ while reducing this search space.

4.5.1. Minimum value of maximum permissible noise

The parameter $r_{\max}$ specifies the permissible noise level at training (synthesis) phase of the GMACAs. To identify the minimum value of $r_{\max}$, extensive experiments are performed to evolve pattern recognizable n-cell GMACAs 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 learnt) is set in the range of 4–10 for different values of n. Fig. 7(a)–(f) demonstrates the percentage of convergence (recognition) for different noise levels. The percentage of convergence (recognition) at a particular noise level r is defined as the ratio of the number of noisy patterns correctly recognized and the total number of noisy patterns with noise level r.

Fig. 7 represents the percentage of recognition for different noise levels ($r_{\max}$) of 1–5 allowed in training (synthesis) phase. The experimental results of Fig. 7(a)–(f) lead to the fact that the GMACAs synthesized with single bit noisy patterns result better convergence at recognition phase irrespective of the noise level r. In the training phase, if we consider patterns with only one bit noise, then the total number of patterns in the state transition table is $k(1 + n)$. In this case the collision of state 0 and 1, as explained in Section 3.2, will be lesser. This leads to higher probability of generation of best fit GMACAs. On the other hand, if we consider noise in more than one bits, then the total number entries in the state transition table is higher. This leads to more collisions. As a result, the evolutionary algorithm of GA fails to arrive at the GMACAs with desired pattern recognition capability. So, the minimum value of $r_{\max}$ is set to 1 at the synthesis (training) phase for which the GMACA based associative memory displays better performance. So, Eq. (2) of Section 3.1 reduces to

$$p = 1 + n, \tag{5}$$

where n represents the number of bits in the patterns to be learnt and p denotes the number of nodes in each sub-graph generated in Phase I of Section 3.1.

4.5.2. Minimum value of maximum permissible height

The parameter $h_{\max}$ specifies the maximum permissible height of each sub-graph at synthesis (training) phase of the GMACAs. To find out the minimum value of $h_{\max}$, we plot the distribution of single bit noisy ($r = 1$) patterns as a function of height h for different values of n and k as the minimum value of $r_{\max}$ is concluded to be 1 at the synthesis phase.

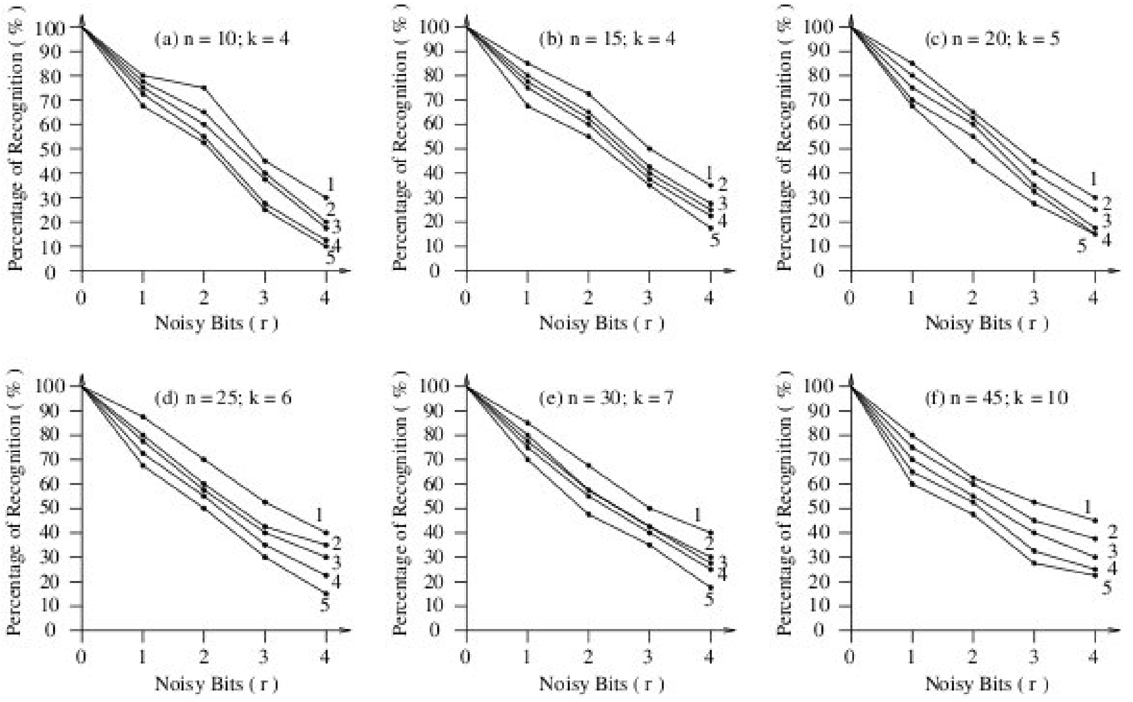


Fig. 7. Percentage of recognition (convergence) with noise level (r).

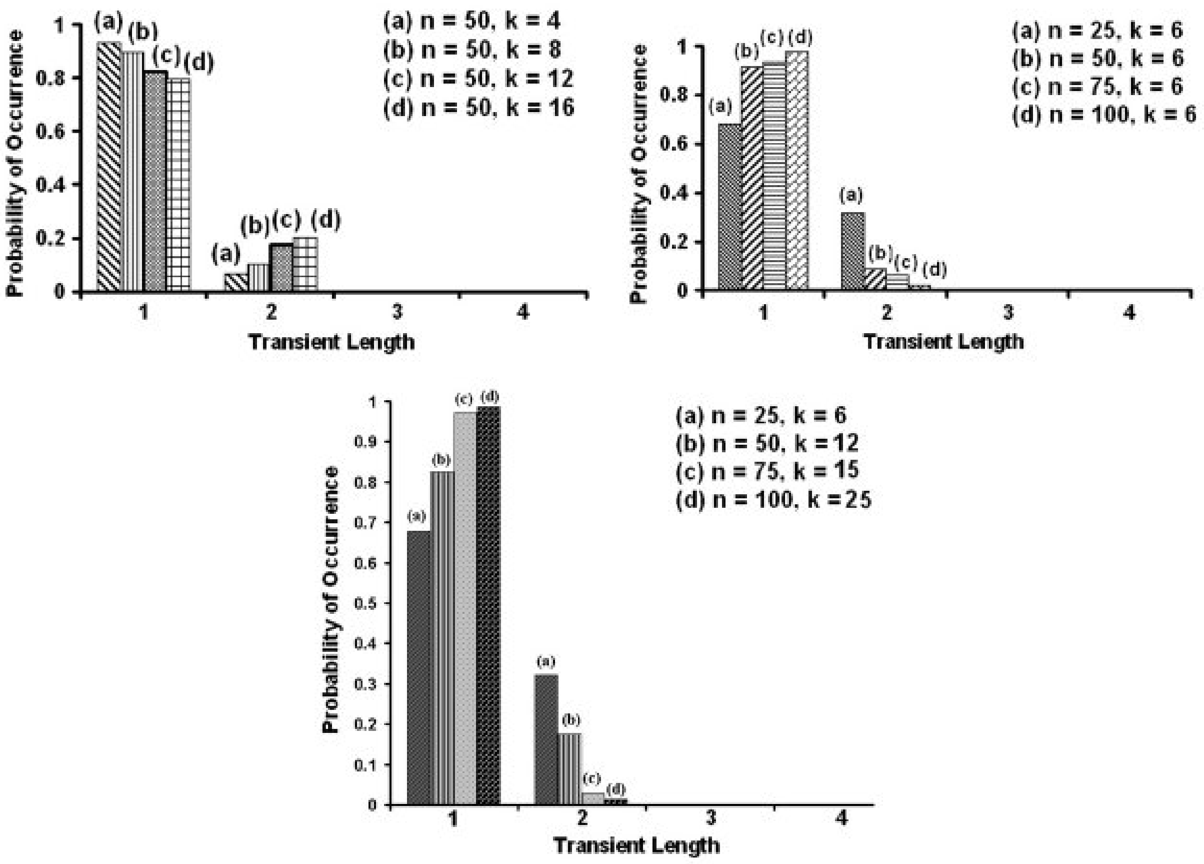


Fig. 8. Distribution of single bit noisy patterns at different heights.

Fig. 8 represents the probability of occurrence of single bit noisy ($r = 1$) patterns as a function of transient length of the single bit noisy patterns, that is, height of the synthesized graph. The results are reported on a set of values of k for a fixed value of n, a set of values of n for a fixed value of k, and a set of n–k pairs. For each case, the result is calculated based on 15 independent runs. From Fig. 8 it is seen that for a fixed value of n, as the value of k increases the probability of occurrence of single bit noisy patterns decreases at height $h = 1$ and increases at $h = 2$. On the other hand, for a fixed value of k, the probability of occurrence increases at $h = 1$ and decreases at $h = 2$ with the increase in n. Also, as the height of the graph (transient length of the noisy patterns corrupted with single bit noise) increases, the probability of occurrence of single bit noisy patterns decreases for a particular value of n and k. The probability of occurrence becomes zero as the height of the graph becomes 3. So, the minimum value of $h_{\max} = 2$, at synthesis phase, for which GMACA behaves as an efficient pattern recognizer. Hence, Eq. (3) of Section 3.1 reduces to

$$G \simeq 1 + (p - 2) = p - 1, \tag{6}$$

where p is the number of nodes in each sub-graph and G denotes the total possible sub-graphs with $h_{\max} = 2$.

4.6. Reduction of search space

This subsection analytically establishes that the proposed GMACA synthesis scheme (Algorithm 2) reduces the search space of the method reported in [9,10] while memorizing a set of n-bit patterns $\mathscr{P}$ with cardinality k. In a 2-state 3-neighborhood CA, all possible CA rules is 256. Hence, for an n-cell non-uniform CA, to memorize a set of n-bit patterns $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$, the total possible search space ($\mathrm{SS_T}$) is

$$\mathrm{SS_T} = 256^n = 2^{8n}. \tag{7}$$

The following theorem formalizes the fact that the search space explored by Algorithm 2 (*Evolving_GMACA*) is insignificant compared to computation of $\mathrm{SS_T}$ (Eq. (7)).

Theorem 1. *The search space explored by Algorithm 2 for the GMACA synthesis scheme is insignificant compared to exhaustive search over the CA rule space.*

Proof. To memorize the pattern set $\mathscr{P}$, we synthesize k number of directed sub-graphs. The number of nodes p in each sub-graph is given by Eq. (2), while Eq. (3) represents the total possible sub-graphs G with p number of nodes. According to Sections 4.5.1 and 4.5.2, the GMACAs behave as efficient pattern recognizer if we consider $p = (1 + n)$ and $G = (p - 1)$ in the synthesis (training) phase. Hence, as per the GA evolution (Algorithm 2), the total search space to memorize the n-bit pattern set $\mathscr{P}$ with cardinality k is given by

$$\mathrm{SS_P} = G^k = n^k. \tag{8}$$

In general, the GMACAs perform well when the number of patterns to be learnt k is equal to 20% of the number of bits n, that is, $k = 0.2n$ [9,10]. Hence, according to Eqs. (7) and (8), the value of $\mathrm{SS_P}$ is being insignificant compared to $\mathrm{SS_T}$ even for large value of n. Hence, the result follows. □

To compare the search space explored by the method reported in [9,10] and Algorithm 2, the term search space ratio (SSR) is introduced next.

Definition 1. The search space ratio (SSR) is defined as the ratio of the search space explored by the algorithm proposed in [9,10] and that of Algorithm 2. That is

$$\mathrm{SSR} = \frac{\mathrm{SS_P}}{\mathrm{SS_T}} = \frac{n^k}{256^n}, \tag{9}$$

where n is the number of bits in a pattern and k is the number of patterns learnt (stored).

The value of SSR is significantly lesser than 1 for a particular value of n and k. For $n = 10$ and $k = 4$, the value of SSR is 8.27×10^{-21}. Also, as the value of n and k increases, the value of SSR decreases with a very high gradient.

5. Analysis of basins of attraction and GMACA rule space

This section provides the characterization of the attractor basins of the desired GMACAs displaying pattern recognition capability and the evolved GMACA rule space. Such characterization is based on the error correcting capability or the basins of attraction of the evolved GMACAs.

5.1. Error correcting capability of GMACAs

In the state space of the GMACA based associative memory, each of the patterns learnt (stored) act as an attractor. A set of GMACA states form the basins of attraction of an attractor. Each state in the basin converges to its attractor. The radius of the basins of attraction is defined as the largest HD within which all patterns converge to the attractor. This gives a measure of the pattern recognition capability for a given associative memory. The theoretical analysis of the basins of attraction proceeds under the following assumptions:

(i) The attractor cycles of the GMACAs are of length 1, that is, we consider only point attractor.
(ii) The GMACAs are designed to recognize noisy patterns within single time step. We have made this assumption for simplification of the theoretical analysis.

According to the above assumptions, if a GMACA is designed to recognize k number of n-bit patterns $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$, each $\mathscr{P}_x$ $(x = 1, 2, \ldots, k)$ must be the single cycle attractor of different attractor basins and a noisy pattern $\acute{\mathscr{P}}_x$ should be correctly recognized as $\mathscr{P}_x$ in single time step. The GMACA is built around 3-neighborhood CA. The single bit noise at the ith position of a noisy pattern $\acute{\mathscr{P}}_x$ derived out of error free pattern $\mathscr{P}_x$ (where $\mathrm{HD}(\mathscr{P}_x, \acute{\mathscr{P}}_x) = 1$) affects only another two positions, $(i-1)$th and $(i+1)$th of $\acute{\mathscr{P}}_x$ in single time step. Hence, to ensure the error correcting capability of the ith cell, its rule should be so designed that the ith cell recovers the correct bit irrespective of the noise on the $(i-1)$th, ith, or $(i+1)$th bit positions. That is, the total number of noisy patterns with single bit noise to be recovered by the rule of the ith cell is $3k$.

Definition 2. The error correcting capability of the ith cell is defined as the ratio of the number of noisy patterns $\acute{N}$ $(0 \leqslant \acute{N} \leqslant 3k)$ with single bit noise correctly recognized and the total number of noisy patterns with single bit noise. So, the error correcting capability of the ith cell is given by

$$\mathrm{ECC} = \frac{\acute{N}}{3k}. \tag{10}$$

In order to analyze the error correcting capability of the GMACAs, we first calculate it for single bit noise, that is, for $\mathrm{HD}(\mathscr{P}_x, \acute{\mathscr{P}}_x) = 1$. The error correcting capability for multiple bit noise can be derived from the analysis of single bit noise. The following theorem establishes this fact.

Theorem 2. *The error correcting capability of the GMACAs at multiple bit noise occurring at a minimum of 2-bit distance can be derived from that of single bit noise.*

Proof. As we consider only 3-neighborhood dependency, the noise at the ith bit of the noisy pattern $\acute{\mathscr{P}}_x$ cannot affect the noise of the $(i+\acute{d})$th bit of $\acute{\mathscr{P}}_x$, where $\acute{d}$ is an integer and $\acute{d} \geqslant 2$. That is, if the multiple bit noise are separated from each other by two or more than two bits, the GMACA based associative memory can recover all these noise at single time step. Each consecutive 3-bits $(i-1, i, i+1)$ recover single bit noise in single time step. In effect, multiple bit noise are recovered in single time step. So, the error correcting capability of the GMACAs at multiple bit noise can be derived from that of single bit noise. □

5.1.1. Error correcting capability with single bit noise

Let the number of distinct 3-neighborhood configurations of the ith cell is m for k number of attractors (patterns to be learnt) $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$. Then, m must satisfy the relation

$$1 \leqslant m \leqslant 2^3. \tag{11}$$

Let $P(m)$ denotes the probability of occurrence of m number of distinct configurations at the ith cell of all attractors $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_x, \ldots, \mathscr{P}_k\}$ and $E(m)$ denotes the error correcting capability of the ith cell with m number of distinct neighborhood configurations. Then, the error correcting capability of k-attractor basins GMACA to recover single bit noise $(r = 1)$ in single time step is given by

$$\mathrm{ECC}(k,1) = \frac{1}{2^{k-1}} + \frac{(2^k - 2)}{2^k} \sum_{m=2}^{y} P(m)E(m), \tag{12}$$

where $y = \min(k, 2^3)$. The detailed explanation and computation of $\mathrm{ECC}(k,1)$ of the GMACAs are reported in [17]. Fig. 9 represents the error correcting capability of k-attractor basins GMACA at single bit noise, that is $\mathrm{ECC}(k,1)$ (Eq. (12)). It depicts that as the value of k (number of attractors) increases, the error correcting capability of the GMACAs decreases. Ultimately, it saturates to 67% for $k \geqslant 8$.

However, as per Theorem 2 if the multiple bit noise are separated by two or more bits distance, the GMAC-A based associative memory can accommodate all these noisy bits in single time step. If a noisy pattern cor-rupted with multiple bit noise satisfies above condition of separation by two or more bit distance, the GMACAs can recognize it correctly. Let the total number of n-bit patterns with r bit noise is $N(n,r)$ where each and every two noisy bits are separated by $\acute{d}$-bit $(\acute{d} \geqslant 2)$ distance. The error correcting capability of n-bit GMACAs with k-attractor basins at multiple bit noise is then given by

$$\mathrm{ECC}(n,k,r) = \frac{N(n,r)}{{}^{n}C_r}\,\mathrm{ECC}(k,1) \text{ and } {}^{n}C_r = \binom{n}{r}, \tag{13}$$

where r is the number of noisy bits. The detailed explanation and computation of $\mathrm{ECC}(n,k,r)$ of the GMA-CAs are reported next.

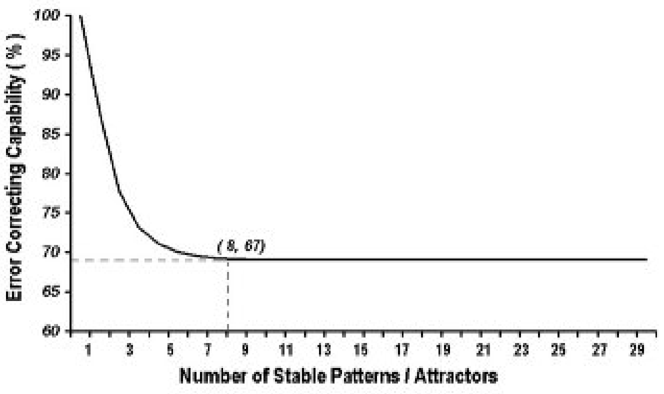


Fig. 9. Error correcting capability of the GMACAs to recover single bit noise in single time step.

5.1.2. Error correcting capability with multiple bit noise

This subsection reports the error correcting capability of the GMACA based associative memory to accommodate multiple bit noise, which gives the measure of the basins of attraction. While the first theorem gives the number of pattern satisfy the desired condition of Theorem 2, the next theorem establishes the error correcting capability at multiple bit noise or the basins of attraction of the GMACAs.

Theorem 3. *The total number of n-bit patterns with r-bit noise, that can be accommodated by the GMACA based associative memory in single time step, is given by*

$$N(n,r) = \binom{n}{r} + \sum_{i=1}^{r-1}(-1)^i\binom{r-1}{i}\sum_{j=0}^{i}\binom{i}{j}\binom{n-i-j}{n-r-j}. \tag{14}$$

Proof. According to Theorem 2, if the multiple bit noise are separated by two or more than two bit distance, the GMACA based associative memory can accommodate all these noisy bits in single time step. So, if a noisy pattern corrupted with multiple bit noise satisfies above condition, the GMACAs can recognize it correctly. Now

$$N(n,1) = n = \binom{n}{1}; \quad N(n,2) = \frac{n(n-1)}{2} - (2n-3) = \binom{n}{2} - \binom{n-1}{n-2} - \binom{n-2}{n-3};$$
$$N(n,3) = \binom{n}{3} - 2\left\{\binom{n-1}{n-3} + \binom{n-2}{n-4}\right\} + \left\{\binom{n-2}{n-3} + 2\binom{n-3}{n-4} + \binom{n-4}{n-5}\right\};$$
$$\cdots$$
$$\text{Hence, } N(n,r) = \binom{n}{r} + \sum_{i=1}^{r-1}(-1)^i\binom{r-1}{i}\sum_{j=0}^{i}\binom{i}{j}\binom{n-i-j}{n-r-j}.$$

Thus, the value of $N(n,r)$ represents the total number of n-bit patterns with r bit noise where each and every two noisy bits are separated by $\acute{d}$-bit ($\acute{d} \geqslant 2$) distance. Hence, the result follows. □

Theorem 4. *The error correcting capability or the basins of attraction of n-bit GMACAs with k-attractor basins is given by*

$$\mathrm{ECC}(n,k,r) = \frac{N(n,r)}{{}^nC_r}\mathrm{ECC}(k,1) \text{ and } {}^nC_r = \binom{n}{r}, \tag{15}$$

where r is the number of noisy bits.

Proof. The total number of n-bit patterns with r-bit noise is given by ${}^nC_r$. So, according to Eq. (12) and Theorem 3, the error correcting capability or the basins of attraction of the GMACAs is given by

$$\mathrm{ECC}(n,k,r) = \frac{N(n,r)}{{}^{n}C_r}\,\mathrm{ECC}(k,1) \text{ and } {}^{n}C_r = \binom{n}{r}.$$

Hence, the proof follows. □

Figs. 10 and 11 represent the basins of attraction in terms of the error correcting capability of the GMACAs recognizing k number of n-bit patterns. The graphs plot the error correcting capability $\mathrm{ECC}(n,k,r)$ denoted by Eq. (13) in the y-axis, while the number of noisy bits r is plotted on the x-axis. From Figs. 10 and 11, it is seen that, for a fixed value of n (number of bits in a pattern) and k (number of attractors), as the value of r (noisy bits) is increased, the error correcting capability of the GMACAs monotonically decreases. For a fixed values of n and r, the function $\mathrm{ECC}(n,k,r)$, as shown in Fig. 10, decreases as the number of attractors (k) increases. Fig. 11 shows that as the number of bits (n) increases, the error correcting capability of the GMACAs also increases for a fixed value of k.

5.2. CA rule analysis

The analysis of the error correcting capability or the basins of attraction of the GMACAs (as detailed in Section 5.1) shows that the CA rules that are effective to design the GMACA based associative memory for

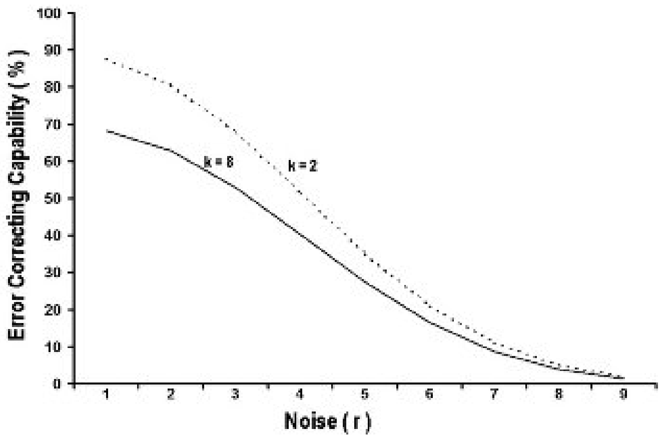


Fig. 10. Error correcting capability of the GMACAs at multiple bit noise for $n = 50$.

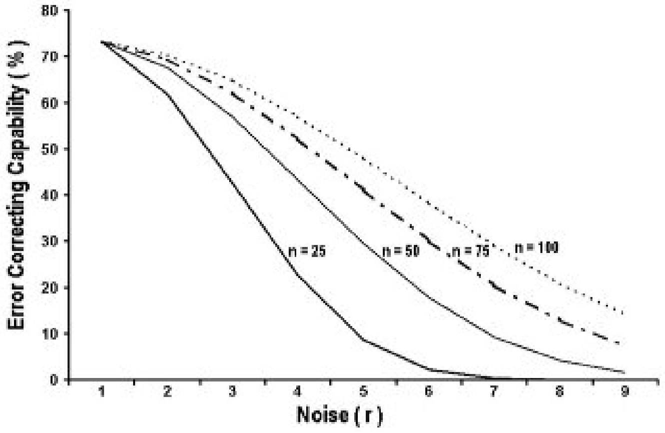


Fig. 11. Error correcting capability of the GMACAs at multiple bit noise for $k = 10$.

pattern recognition must satisfy the following conditions. If out of the total eight possible neighborhood configurations of a CA rule, for two configurations C_1 and C_2, the present state of the ith cell is 1, then

(i) $\mathrm{HD}(C_1, C_2) = 1$;
(ii) the next state for both C_1 and C_2 must be 1; and
(iii) the next state for $\acute{C}_1$ and $\acute{C}_2$ must be 0, where $\mathrm{HD}(C_1, \acute{C}_1) = 3$ and $\mathrm{HD}(C_2, \acute{C}_2) = 3$.

So, the GMACA rules that are effective for pattern recognition must contain $\bar{r}$ number of ones as the next state, where $2 \leqslant \bar{r} \leqslant 6$ (except Rule 0 and Rule 255). If $N(\bar{r})$ denotes the number of rules with $\bar{r}$ number of ones as the next state which satisfy the above conditions, then

$$N(\bar{r}) = \sum_{i=2}^{4} \left\{ \binom{4}{i} - 2\binom{2}{i} \right\} \left\{ \sum_{j=1}^{\bar{r}-i} \binom{4-i}{\bar{r}-i-j} \sum_{k=j}^{i-2} \left\{ 2j + \binom{4-i}{\bar{r}-i} \right\} \right\}. \tag{16}$$

Hence, the probability $P_{\bar{r}}$ of a CA rule (with $\bar{r}$ number of 1's as the next state) to be effective to design the GMACA based associative memory for pattern recognition is given by

$$P_{\bar{r}} = \frac{N(\bar{r})}{T(\bar{r})}, \quad \text{where } T(\bar{r}) = \binom{8}{\bar{r}} \tag{17}$$

and $T(\bar{r})$ represents the total number of rules with $\bar{r}$ number of 1's.

As the value of $\bar{r}$ (number of ones in a rule) is increased from 1 to 4, the value of $P_{\bar{r}}$ increases, while $\bar{r}$ is shifted from 4 to 7, the value of $P_{\bar{r}}$ decreases. So, the value of $P_{\bar{r}}$ attains its maximum value at $\bar{r} = 4$. Fig. 12 represents the probability of being an effective CA rule with $\bar{r}$ number of ones as a next state with respect to $\bar{r}$ (number of ones in a rule). To analyze the GMACA rules in terms of number of ones and zeros as the next state, we introduce the term homogeneity (H) of the CA rule, which is defined next.

Definition 3. If $\bar{r}$ is the number of ones of a CA rule, the rule must contain $(8 - \bar{r})$ number of zeros as the next state. Then, the homogeneity H of that rule is given by

$$H = \left| 1 - \frac{\bar{r}}{4} \right|. \tag{18}$$

According to Eq. (18), if the value of $\bar{r}$ of a rule is either 0 or 8, the value of H of that rule is 1 (maximum); whereas if $\bar{r} = 4$, the value of H of the corresponding rule is 0 (minimum). Thus, if most of the neighborhood configurations of a CA rule map to same next state, either 0 or 1, the rule becomes more homogeneous.

Theorem 5. *The probability of a CA rule to be effective for pattern recognition decreases with the increase in its homogeneity value.*

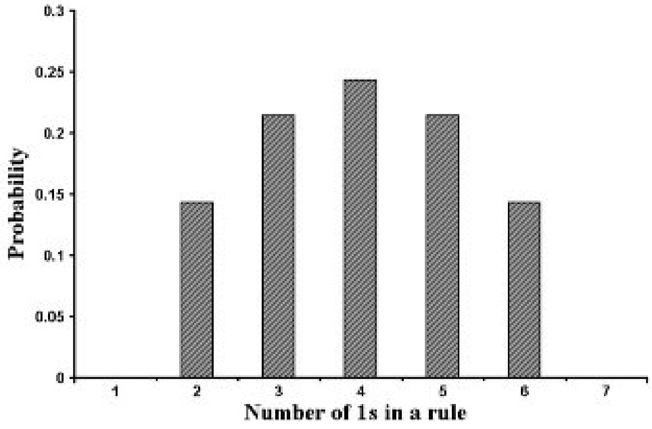


Fig. 12. Probability of being an effective CA rule for pattern recognition with respect to number of ones.

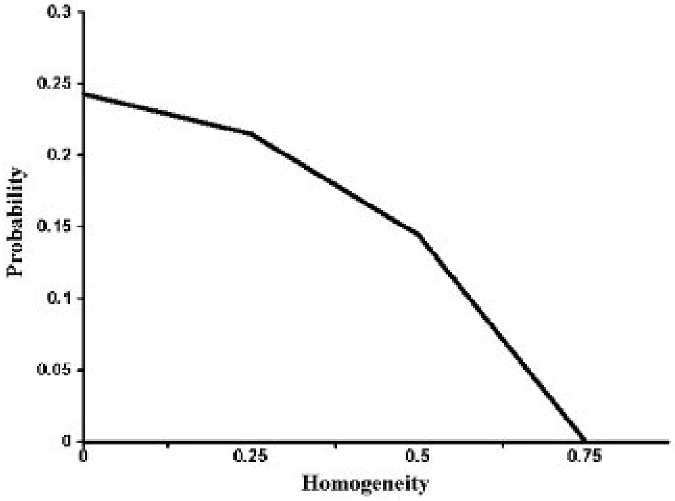


Fig. 13. Probability of being an effective CA rule for pattern recognition with respect to homogeneity H.

Proof. As per Eqs. (17) and (18), as the value of $\tilde{r}$ is increased from 1 to 4, the probability $P_{\tilde{r}}$ of being an effective CA rule (with $\tilde{r}$ number of ones) increases, while its homogeneity H decreases. When the value of $\tilde{r}$ is shifted from 4 to 7, the value of $P_{\tilde{r}}$ decreases, while the value of H of the CA rule increases. So, the value of $P_{\tilde{r}}$ of a CA rule decreases with the increase in its homogeneity H. □

Fig. 13 represents the probability $P_{\tilde{r}}$ of the CA rules with respect to their homogeneity H. It depicts that as the homogeneity increases, the value of $P_{\tilde{r}}$ of a CA rule decreases. That is, the CA rules having close to an equal number of 0's and 1's as the next state tend to be much better pattern recognizers than the CA rules having a relatively high degree of homogeneity. In CA with more homogeneous rules, the states of too many cells freeze early in the dynamics. The tendency of freezing increases as the rule becomes more and more homogeneous. Hence, the CAs formed with such rules don't acquire the capacity to accommodate noise.

6. Performance analysis

In this section, extensive experiments are performed based on randomly generated pattern set for different values of n (number of bits in a pattern) and k (number of patterns to be learnt) to analyze the performance of the GMACA based associative memory model. The detailed experimental results reported next validate the analytical framework of the CA based associative memory discussed in Section 5 as well as the GA framework that is set to arrive at the desired GMACAs with feasible computation. Also, the quality of the desired solution can be gauged from the basins of attraction of the evolved GMACAs, evolution time to synthesize the desired GMACAs, recognition complexity, and the distribution of the CA rules.

6.1. Basins of attraction

To measure the size and shape of the basins of attraction of the evolved GMACAs, a process is chosen similar to that used in [8], which is outlined next:

(i) Choose a stable state or attractor $\mathscr{P}_x (x = 1, 2, \ldots, k)$.
(ii) Choose some initial noise value $r = r_0$.
(iii) Let the set A be all states of HD r from the attractor $\mathscr{P}_x$.
(iv) Sample 100 states from A.
(v) Calculate how many of these states are attracted to the attractor $\mathscr{P}_x$. Denote this number $t_x(r)$.
(vi) Increment the noise value r by a suitable amount and repeat from (iii).
(vii) Repeat for each stored pattern x, where $x = 1, 2, \ldots, k$.

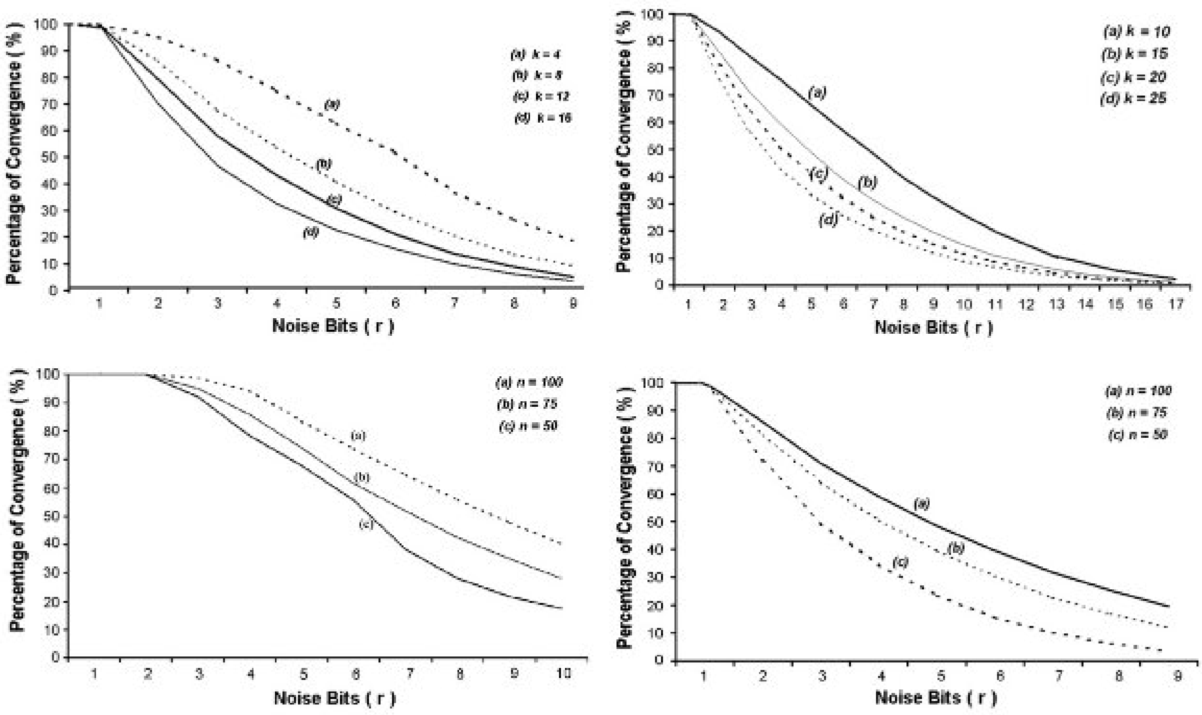


Fig. 14. Percentage of convergence (recognition) with noisy bits (r) (i) $n = 50$ and 100; (ii) $k = 5$ and 15.

The value of $t_x(r)$ gives an estimate for the percentage of patterns or states distance r from the attractor $\mathscr{P}_x$, which are attracted to the fixed point $\mathscr{P}_x$. It also gives a measure of the skew of the basins of attraction. Fig. 14 represents the performance of recognition or convergence at different noise values per bit. All these figures show the percentage of convergence as a function of noise with respect to stored patterns (k) and the number of bits of input patterns (n). Computation shown is for a fixed HD between patterns stored. It is seen that the convergence rate drops almost linearly with the amount of noise in stored pattern. In first two graphs of Fig. 14, it is seen that the convergence rate also reduces as the value of k increases for a fixed noise value. From the results reported in last two graphs of Fig. 14, it is seen that for a fixed value of k, the convergence rate increases as the value of n increases. The recognizer performs very well at $n = 100$, but accommodates noise rather poorly at $n = 50$ for $k = 5$ and 15. All the results reported in Fig. 14 closely match with that of theoretical formulation noted in Section 5.1.

Fig. 15 presents the percentage of recognition or convergence at different noise levels both for the method reported in [9,10] and the proposed method. The results are reported for different values of n and k. All the results reported in Fig. 15 establish the fact that the performance of the proposed method at a particular noise level is significantly higher compared to that of the method reported in [9,10].

6.2. Evolution time

The evolution time of the desired GMACAs are reported here for different values of n and k, both for the method reported in [9,10] and Algorithm 2 for the GMACA evolution. All the algorithms are implemented in C language and run in Linux (version 2.2.6) environment having machine configuration Pentium II (i686), 400 MHz, 512 KB cache, and 132 MB RAM.

Table 1 represents the evolution time and the number of generations required to converge the GA to synthesize the desired GMACAs in both cases. Columns I and II of Table 1 represent the different CA size (n) and the number of patterns to be learnt (k), respectively. While Column III depicts the maximum number of generations required to find out the best possible GMACA configuration by the GA and evolution time, respectively for the method reported in [9,10], Column IV shows that of the proposed scheme. The entries '*' in

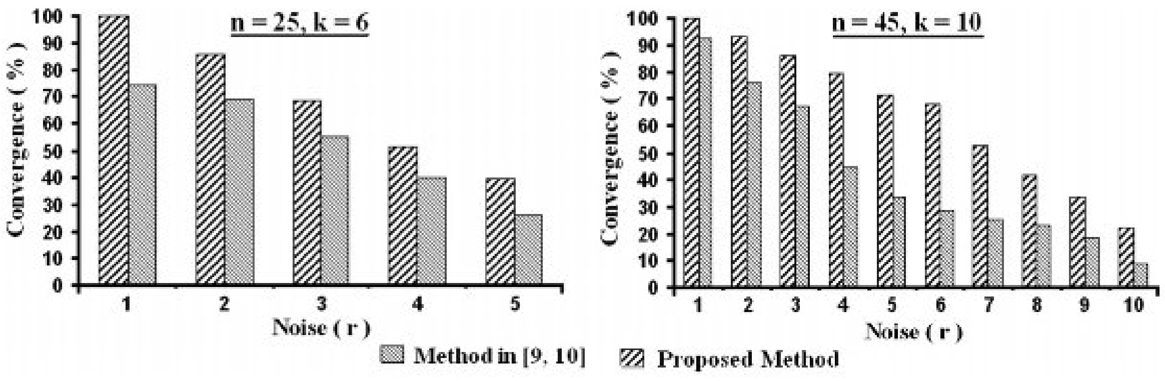


Fig. 15. Percentage of convergence (recognition) with noisy bits for $n = 25$, $k = 6$ and $n = 45$, $k = 10$.

Table 1
Evolution time to synthesize the GMACAs of the method in [9,10] and proposed scheme

CA size (n)	Value of k	Method in [9,10]		Proposed scheme	
		# Gene[a]	Time (s)	# Gene[a]	Time (s)
10	4	102	18.18	35	12.98
15	4	113	21.37	42	16.20
20	5	132	24.87	53	21.73
25	6	171	31.17	61	26.91
30	7	193	37.27	74	32.02
35	8	228	42.73	79	34.28
40	10	263	55.47	92	41.91
45	10	327	78.67	94	44.05
50	12	*	*	98	46.79
55	12	*	*	98	51.34
60	13	*	*	103	56.09
65	15	*	*	117	82.03
70	15	*	*	122	97.16
75	16	*	*	137	128.29
80	18	*	*	145	161.38
85	18	*	*	161	178.63
90	20	*	*	172	212.01
95	20	*	*	179	265.64
100	23	*	*	187	316.89
...	...	...	...	...	...

Column III of Table 1 signify that the GA in [9,10] does not converge within 1000 generations. All the results reported in Table 1 proves the superiority of the proposed scheme over the scheme of [9,10].

6.3. Recognition complexity

In the recognition phase, when a pattern comes as an input (without or with distortion due to noise), the GMACA is run with the 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 the given input pattern to the correct attractor. Hence, the cost of computation for entire recognition or identification process is constant.

Fig. 16 represents the time taken to recognize a noisy pattern for different values of n and k in terms of the transient length or the number of iterations. The results are reported for both the method reported in [9,10] and the proposed algorithm. All the results reported in Fig. 16 ensure that the time taken to recognize a noisy pattern is lesser in the proposed scheme compared to that of method reported in [9,10].

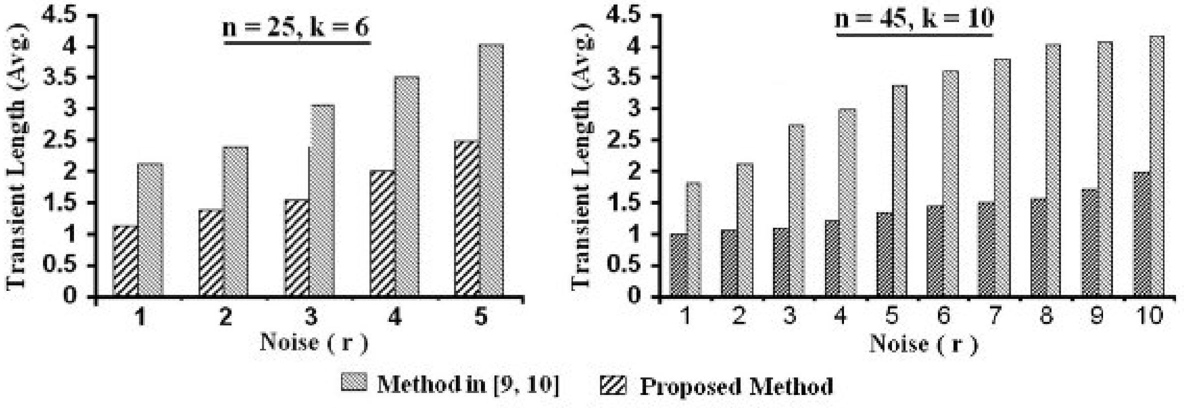


Fig. 16. Average iteration with noisy bits for $n = 25$, $k = 6$ and $n = 45$, $k = 10$.

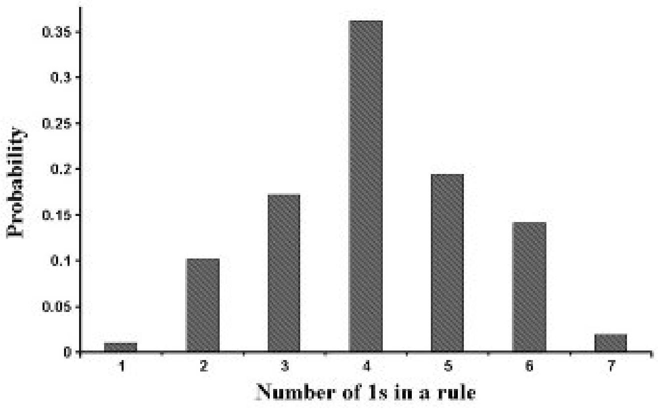


Fig. 17. Probability of being an effective CA rule with respect to number of ones.

6.4. Distribution of CA rule

The probability of being an effective CA rule is studied extensively in terms of the number of 1's in a rule. The probability distribution of different CA rules derived from evolutionary algorithm of the GA is shown in Fig. 17. The distributions reported in Figs. 12 and 17 confirm that the rule space theoretically derived is a subset of the CA rules arrived at through the GMACA evolution with the GA.

7. Conclusion

The paper contributes to the research in both the area of CAs and that of pattern recognition. One of the major objectives of the current research is to explore the computational capability of the non-uniform CAs in the field of pattern recognition. The pattern recognizer is designed with a special class of non-uniform CAs, termed as the GMACAs. The exponential growth of the search space of the non-uniform CAs is addressed through a reverse engineering technique. The GA based evolutionary algorithm is coupled with this reverse engineering technique to synthesize the desired GMACA configurations. The scheme allows the user to perform directed search for the desired GMACAs. Also, it significantly reduces the search space of the GMACA based associative memory.

Moreover, the characterization of the basins of attraction of the GMACA based associative memory is introduced. A number of fundamental results are reported to characterize the state transition behavior of the GMACAs. Such results enable visualization of non-uniform CA evolution. Using the concept of homogeneity introduced in this paper the desired CA rules are analyzed. Theoretical formulation supported with extensive experimental results reported in this paper confirms that the GMACA based associative memory can be employed as an excellent pattern recognizer. Hence, the major contributions of this paper can be summarized as follows:

– the evolved GMACAs have attractor cycle of length 1, each attractor signifying a single cycle attractor;
– an efficient formulation of the GA is coupled with a reverse engineering technique to synthesize the desired GMACA configurations;
– the proposed design significantly reduces the search space of the GMACA based associative memory reported in [9,10];
– theoretical formulation of the basins of attraction of the GMACAs, as well as extensive experimental results reported in this paper, establish the GMACA as an efficient and cost-effective solution for pattern recognition problem; and
– the characterization of the GMACA rule space validates the experimental observations reported in [9].

Thus, applying a non-uniform CA based model to pattern recognition task and designing a GA based reverse engineering technique for this architecture, the application areas of this powerful computational model are extended as well as contributed to the research on learning with the CAs.

Acknowledgements

The authors would like to thank anonymous referees for providing helpful comments and valuable criticisms on the original version of the manuscript which have greatly improved the presentation of paper.

References

[1] J. Buhmann, R. Divko, K. Schulter, Associative memory with high information content, Physical Review A 39 (5) (1989) 2689–2692.
[2] M. Chady, R. Poli, Evolution of cellular-automaton-based associative memories, in: Second On-Line World Conference on Soft Computing in Engineering Design and Manufacturing, 1997, pp. 1–8.
[3] J.-C. Chen, C.-M. Yeh, J.-E. Tzeng, Pattern differentiation of glandular cancerous cells and normal cells with cellular automata and evolutionary learning, Expert Systems with Applications 34 (1) (2008) 337–346.
[4] R. Das, M. Mitchell, J.P. Crutchfield, A Genetic Algorithm Discovers Particle Based Computation in Cellular Automata, Parallel Problem Solving from Nature, Springer-Verlag, 1994, pp. 244–353.
[5] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Ph.D. thesis, University of Michigan, 1975.
[6] P.P.B. de Oliveira, J.C. Bortot, G.M.B. Oliveira, The best currently known class of dynamically equivalent cellular automata rules for density classification, Neurocomputing 70 (1–3) (2006) 35–43.
[7] R. Dewri, N. Chakraborti, Simulating recrystallization through cellular automata and genetic algorithms, Modelling and Simulation in Materials Science and Engineering 13 (2005) 173–183.
[8] B.M. Forrest, Content-addressability and learning in neural networks, Journal of Physics A: Mathematical and General 21 (1988) 245–255.
[9] N. Ganguly, P. Maji, B.K. Sikdar, P.P. Chaudhuri, Generalized multiple attractor cellular automata (GMACA) for associative memory, International Journal of Pattern Recognition and Artificial Intelligence 16 (7) (2002) 781–795.
[10] N. Ganguly, P. Maji, B.K. Sikdar, P.P. Chaudhuri, Design and characterization of cellular automata based associative memory for pattern recognition, IEEE Transactions on System, Man and Cybernetics, Part B 34 (1) (2004) 672–678.
[11] D.E. Goldberg, Genetic Algorithms in Search, Optimizations, and Machine Learning, Morgan Kaufmann, 1989, pp. 27–57 (Chapter 2).
[12] J.H. Holland, Adaptation in Natural and Artificial Systems, The MIT Press, 1992, pp. 89–120 (Chapter 6).
[13] E. Jen, Invariant strings and pattern recognizing properties of 1D CA, Journal of Statistical Physics 43 (1986) 243–265.
[14] C.G. Langton, Self-reproduction in cellular automata, Physica D 10 (1984) 135–144.
[15] K. Laurio, F. Linaker, A. Narayanan, Regular biosequence pattern matching with cellular automata, Information Sciences 146 (1–4) (2002) 89–101.
[16] P. Maji, Cellular automata evolution for pattern recognition, Ph.D. thesis, Jadavpur University, India, 2005.
[17] P. Maji, N. Ganguly, P.P. Chaudhuri, Error correcting capability of cellular automata based associative memory, IEEE Transactions on System, Man and Cybernetics, Part A 33 (4) (2003) 466–480.
[18] M. Mitchell, P.T. Hraber, J.P. Crutchfield, Revisiting the edge of chaos: evolving cellular automata to perform computations, Complex Systems 7 (1993) 89–130.

[19] J.H. Moore, L.W. Hahn, Multilocus pattern recognition using one-dimensional cellular automata and parallel genetic algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2001, p. 1452.
[20] K. Morita, S. Ueno, Parallel generation and parsing of array languages using reversible cellular automata, International Journal of Pattern Recognition and Artificial Intelligence 8 (1994) 543–561.
[21] N. Prosperini, D. Perugini, Application of a cellular automata model to the study of soil particle size distributions, Physica A: Statistical Mechanics and its Applications 383 (2) (2007) 595–602.
[22] R. Raghavan, Cellular automata in pattern recognition, Information Sciences 70 (1–2) (1993) 145–177.
[23] T.D. Rane, R. Dewri, S. Ghosh, K. Mitra, N. Chakraborti, Modeling the recrystallization process using inverse cellular automata and genetic algorithms: studies using differential evolution, Journal of Phase Equilibria and Diffusion 26 (2005) 311–321.
[24] A. Rosenfeld, A.Y. Wu, T. Dubitzki, Fast language acceptance by shrinking cellular automata, Information Sciences 30 (1) (1983) 47–53.
[25] D. Shuai, Y. Dong, Q. Shuai, A new data clustering approach: generalized cellular automata, Information Systems 32 (7) (2007) 968–977.
[26] M. Sipper, Co-evolving non-uniform cellular automata to perform computations, Physica D 92 (1996) 193–208.
[27] M. Sipper, Evolution of Parallel Cellular Machines The Cellular Programming Approach, Springer-Verlag, Heidelberg, 1997, pp. 73–100 (Chapter 4).
[28] S.K. Tan, S.-U. Guan, Evolving cellular automata to generate nonlinear sequences with desirable properties, Applied Soft Computing 7 (3) (2007) 1131–1134.
[29] T. Toffoli, Reversible computing, in: J.W. De Bakker, J. Van Leeuwen (Eds.), Automata, Languages, and Programming, 1980, pp. 632–644.
[30] T. Toffoli, N. Margolus, Cellular Automata Machines, The MIT Press, Cambridge, Massachusetts, 1997.
[31] M. Tomassini, M. Perrenoud, Cryptography with cellular automata, Applied Soft Computing 1 (2) (2001) 151–160.
[32] G. Vichniac, Simulating physics with cellular automata, Physica D 10 (1984) 96–115.
[33] S. Wolfram, Cellular Automata and Complexity, Westview Press, 2002, pp. 159–202 (Chapter 4).