Theory and Application of Cellular Automata For Pattern Classification

Pradipta Maji*[*] and Chandrama Shaw
Department of Computer Science and Technology,
Bengal Engineering College (D U), Howrah, India 711103,
pradipta;cshaw@cs.becs.ac.in

Niloy Ganguly
Technical University of Dresden,
High Performance Computing Centre,
Zellescher Weg 12, Willers-Bau A 115, D-01069 Dresden,
niloy@zhr.tu-dresden.de.

Biplab K. Sikdar and P. Pal Chaudhuri
Department of Computer Science and Technology,
Bengal Engineering College (D U), Howrah, India 711103,
biplab;ppc@cs.becs.ac.in

Abstract. This paper presents the theory and application of a high speed, low cost pattern classifier. The proposed classifier is built around a special class of sparse network referred to as Cellular Automata (CA). A specific class of CA, termed as Multiple Attractor Cellular Automata (MACA), has been evolved through Genetic Algorithm (GA) formulation to perform the task of pattern classification. The versatility of the classification scheme is illustrated through its application in three diverse fields - data mining, image compression, and fault diagnosis. Extensive experimental results demonstrate better performance of the proposed scheme over popular classification algorithms in respect of memory overhead and retrieval time with comparable classification accuracy. Hardware architecture of the proposed classifier has been also reported.

Keywords: Cellular Automata (CA), Multiple Attractor Cellular Automata (MACA), Pattern Classification, Genetic Algorithm (GA), Data Mining, Image Compression, Fault Diagnosis.

[*] Address for correspondence: Department of Computer Science and Technology, Bengal Engineering College (D U), Howrah, India 711103

1. Introduction

This paper presents the theory and application of a pattern classifier built with the sparse network of Cellular Automata (CA). Following scenario provided the motivation to undertake this research.

The inter-networked society of cyber-age has been experiencing an explosion of data in diverse fields. However, meaningful interpretation of this voluminous data is becoming increasingly difficult. Efficient techniques for extraction of knowledge from large database is a basic necessity of the current age. Data classification provides a solution to this problem.

Data classification is the process of identifying common properties among a set of objects in a database. It classifies the objects into different classes. A sample set from the database, each member belonging to one of the predefined classes, is used to train the model. Subsequent to training, the model performs the task of prediction. The prediction phase outputs the desired class in which the input data belongs [16].

The essential prerequisites of designing the classifier for current information age are high throughput and low storage requirements. Further, low cost hardwired implementation of the scheme is becoming a very important criterion for on line real time applications. The conventional techniques developed for classification - Bayesian Classification [9], Neural Network [17], Genetic Algorithm [14], Decision Tree [6, 22, 28, 32] are too complex to meet such requirements.

In this background, design of pattern classifier based on a special class of Cellular Automata (referred to as Multiple Attractor Cellular Automata (MACA)) has been explored in a number of papers [7, 12, 24, 34, 35]. Design of a two class classifier have been proposed in [12]. In the current paper we consolidate and refine the design approach while providing a detailed insight into the applications of the CA based pattern classifier in diverse fields. Experimental results of data classification in the fields of data mining, image compression, and fault diagnosis have established the superiority, versatility, and scalability of the proposed scheme. The major contributions of this paper can be summarized as follows.

- The special class of CA referred to as Multiple Attractor CA (MACA) is employed to design the proposed pattern classifier. MACA acts as an implicit memory; consequently the memory overhead of the proposed scheme has come down significantly.

- The complexity of classification algorithm is linear.

- Excellent classification accuracy of MACA in diverse applications like - data mining, image compression, fault diagnosis in electronic circuits, etc., have been established through extensive experimentation reported in this paper.

- The desired structure of MACA for a particular dataset is obtained through Genetic Algorithm (GA) formulation that leads to significant reduction of search space.

- The classifier employs the simple computing model of 3-neighborhood linear CA. The simple, regular, modular and local neighborhood sparse network of CA suits ideally for low cost VLSI implementation [8].

In the above background we proceed to report CA preliminaries including MACA (Multiple Attractor CA) fundamentals in *Section 2.* This is followed by a new linear operator termed as Dependency Vector/String used to characterize MACA. Design of MACA based classifier is next presented in *Section*

4 along with the Genetic Algorithm (GA) formulation. In order to validate the theoretical formulation of the proposed classifier, the experimental results on three application areas - data mining, image compression, and fault diagnosis are reported in *Sections 5*, *6*, and *7* respectively. Finally, the *Section 8* reports the hardware architecture of the proposed CA based pattern classifier.

2. Cellular Automata (CA)

A Cellular Automaton (CA) can be viewed as an autonomous finite state machine (FSM) consisting of a number of cells [23]. In a 3-neighborhood dependency, the next state $q_i(t+1)$ of a cell is assumed to be dependent only on itself and on its two neighbors (left and right), and is denoted as

$$q_i(t+1) \ = \ f(q_{i-1}(t), q_i(t), q_{i+1}(t))$$

where $q_i(t)$ represents the state of the i^{th} cell at t^{th} instant of time. *'f'* is the next state function and referred to as the rule of the automata. The decimal equivalent of the next state function, as introduced by Wolfram [36], is the rule number of the CA cell. For example,

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

where $\oplus$ (XOR function) denotes modulo-2 addition. Since *'f'* is a function of 3 variables, there are 2^{2^3} i.e., 256 possible next state functions (rules) for a CA cell. In this paper, we have used hybrid (also referred to as non-homogeneous) null boundary CA. A hybrid CA employs different rules to different cells, while in a null boundary CA, left/right neighbor of the leftmost/rightmost terminal cell is connected to logic 0. Out of 256 rules there are only 7 rules with XOR logic (Table 1). The CA employing XOR rule is referred to as linear CA. The pattern classifier proposed in this paper employs linear CA. The theory of linear CA and its applications in diverse fields have been extensively dealt with in [8].

Table 1. Linear CA rules

Rule No.	Next State Logic Function
Rule 60	$q_i(t+1) \ = \ q_{i-1}(t) \oplus q_i(t)$
Rule 90	$q_i(t+1) \ = \ q_{i-1}(t) \oplus q_{i+1}(t)$
Rule 102	$q_i(t+1) \ = \ q_i(t) \oplus q_{i+1}(t)$
Rule 150	$q_i(t+1) \ = \ q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$
Rule 170	$q_i(t+1) \ = \ q_{i+1}(t)$
Rule 204	$q_i(t+1) \ = \ q_i(t)$
Rule 240	$q_i(t+1) \ = \ q_{i-1}(t)$

2.1. Characterization of Linear CA

The global state S_t of an n-cell CA at t^{th} instant of time is a string of n binary symbols. The next state S_{t+1} of the CA is given by: $S_{t+1} = T.S_t$. T is an $n \times n$ characteristic matrix [8] (also referred to as

Dependency Matrix), where

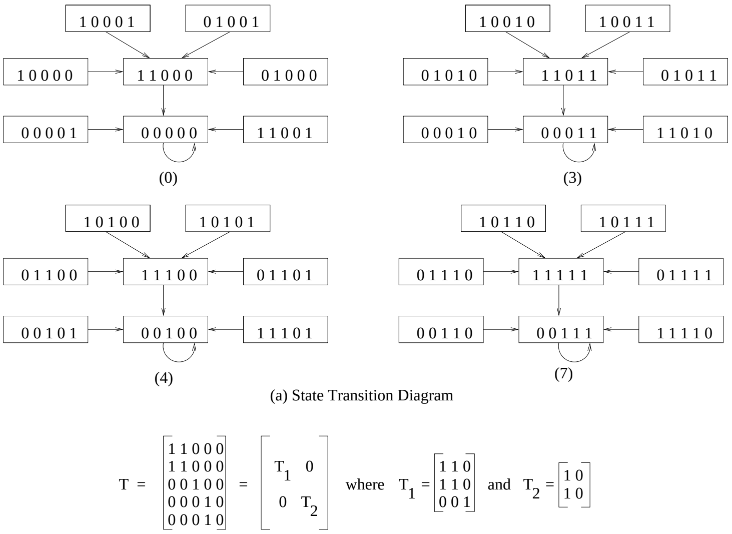
$$T_{ij} = \begin{cases} 1, & \text{if next state of } i^{th} \text{ cell depends on present} \\ & \text{state of } j^{th} \text{ cell; } i,\ j\ =\ 1,\ 2,\ ...,\ n \\ 0, & \text{otherwise.} \end{cases}$$

If we restrict to 3-neighborhood dependency, then T_{ij} can have non-zero values only for $j = (i-1)$, i, and $(i+1)$. Thus, T becomes a tri-diagonal matrix. The polynomial $\phi(x)$ of which T is a root is referred to as the characteristic polynomial of the CA. The characteristic polynomial is derived from T by calculating det$(T + Ix)$ [8, 13].

If all the states in the state transition diagram of a CA lie in some cycles, it is a group CA, whereas a non-group CA state transition graph has both cyclic and non-cyclic states (Fig 1). For a group CA, det$[T] \neq 0$; while for the non-group CA, det$[T] = 0$. In this paper we have employed a special class of non-group CA referred to as Multiple Attractor CA (MACA) for designing the classifier.

2.2. Multiple Attractor Cellular Automata (MACA)

The state transition graph of an MACA consists of a number of cyclic and non-cyclic states. The set of non-cyclic states of an MACA forms inverted trees rooted at the cyclic states. The cyclic states with self loop are referred to as attractors. Fig 1 depicts the state transition diagram of a 5-cell MACA with its rule vector as $< 102, 60, 204, 204, 170 >$. The i^{th} cell (i = 1 to 5) employs the rule specified by the i^{th} element of the rule vector. The corresponding dependency, as specified in Table 1, gets reflected by the i^{th} row of the T matrix. The four cyclic states $\{$00000(0), 00011(3), 00100(4), 00111(7)$\}$ are referred to as attractors and the corresponding inverted tree rooted on α (α = 0, 3, 4, 7) as α-basin.



(a) State Transition Diagram

(b) Dependency Matrix

Figure 1. State transition diagram of a 5-cell MACA with Dependency Matrix (T)

The detailed characterization of MACA is available in [8, 11]. A few fundamental results for an n-cell MACA having k number of attractor basins is next outlined. Each of these has been illustrated under *Example 2.1* that follows.

- *Result I:* The characteristic polynomial of an n-cell MACA with k number of attractor basins is ($x^{n-m} \cdot (1+x)^m$), where $m = log_2(k)$.

- *Result II:*

  Definition 2.1. An m-bit field of an n-bit pattern set is said to be pseudo-exhaustive if all possible 2^m patterns appear in the set.

  Theorem 2.1. [8] In an n-cell MACA with $k = 2^m$ attractors, there exists *m-bit* positions at which the attractors generate pseudo-exhaustive 2^m patterns.

- *Result III:* The pseudo-exhaustive field (PEF) of an attractor provides the pointer to the class of states in the attractor basin. In order to identify the class of a state $\mathcal{P}$, the MACA is initialized with $\mathcal{P}$ and operated for maximum of d (depth) number of cycles till it reaches an attractor. Next, the PEF bits can be extracted (as noted in [8]) to identify the class of $\mathcal{P}$. In general, depth d is defined as the number of time steps an MACA needs to reach an attractor state when it is initialized with a non-reachable state as seed.

- *Result IV:*

  Theorem 2.2. [8] The 0-basin of an n-bit MACA with 2^m attractors forms a vector subspace of dimension $(n-m)$.

- *Result V:* A tri-diagonal Dependency Matrix (T) corresponding to the characteristic polynomial ($x^{n-m} \cdot (1+x)^m$) can be obtained from m number of Dependency Matrices T_is ($i = 1, 2, \cdots, m$) arranged in Block Diagonal Form (Fig 2), where each T_i corresponds to the characteristic polynomial ($x^{n_i - 1} \cdot (1+x)$) and $n_1 + n_2 + \cdots + n_m = n$ [8, 12].

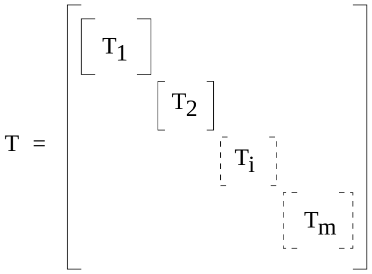


Figure 2. $T_1, T_2, \cdots$, etc. in Block Diagonal Form. T_i (T_j) has no dependency on T_j (T_i)

Example 2.1. The example MACA of Fig 1 is used to illustrate the *Results I* to *V*.

- It is a 5-cell MACA having 4-attractors and depth (d) of the MACA is 2.

- *Result I:* The characteristic polynomial is $(x^3 \cdot (1+x)^2)$. Therefore, m=2, k = number of attractors = 2^m = 4.

- *Result II:* In Fig 1, 3rd and 4th bit positions constitute the PEF.

- *Result III:* Consider a state $\{10001\}$ whose class is to be identified. The MACA of Fig 1 is initialized with $\{10001\}$ and operated for 2 cycles (as depth d = 2) till it reaches an attractor $\{00000\}$. The PEF bits $\{00\}$ of the attractor points to the class of $\{10001\}$.

- *Result IV:* In the 0-basin, (i) the all zero vector is always present; (ii) the modulo-2 sum of any two states in the 0-basin lies in the 0-basin only. These two properties make the 0-basin a vector subspace.

- *Result V:* The Dependency Matrix (T), as indicated in Fig 1(b) with characteristic polynomial $(x^3 \cdot (1+x)^2)$, can be obtained from two matrices T_1 and T_2 by Block Diagonal Form of Fig 2, where T_1 and T_2 correspond to characteristic polynomials $(x^2 \cdot (1+x))$ and $(x \cdot (1+x))$ respectively.

Thus, an MACA acts as an implicit memory. The states of its basin have low hamming distance (HD) among themselves [12] and the basin gets identified by the PEF of the attractor.

2.3. MACA - As A Pattern Classifier

An n-bit MACA with k-attractor basins can be viewed as a natural classifier. It classifies a given set of patterns into k number of distinct classes, each class containing the set of states in the attractor basin. The following example illustrates an MACA based two class pattern classifier.

Example 2.2. Let, the MACA of Fig 1 be employed to classify patterns into two classes (say I and II), where Class I is represented by the states of one set of attractor basins - say [I]= $\{$00100 and 00111$\}$, while Class II is represented by the states in rest of the basins - $\{$00000 and 00011$\}$. All the patterns in the attractor basins [I] belong to Class I while rest of the patterns belong to Class II. As per the *Theorem 2.1,* the pseudo-exhaustive field (PEF) will identify the class of the patterns uniquely. The PEF yields the address of the memory that stores the class information. Therefore, Class I attractors yield the memory address $\{$10, 11$\}$, while Class II will be identified by the memory address $\{$00, 01$\}$ (Fig 3). To identify the class of an input pattern $\mathcal{P}$, the MACA is loaded with $\mathcal{P}$ and operated till it reaches an attractor state. The PEF of the attractor points to the memory location that stores the class information of $\mathcal{P}$.

An attractor basin of an MACA, as reported by Ganguly in [12], has patterns close to each other in respect of hamming distance (HD). Consequently, patterns having low HD between them have high probability of getting covered by lesser number of attractor basins. If $\mathcal{P}$ is a pattern of a class with which the classifier has been trained - that is, MACA has been designed and $\tilde{\mathcal{P}}$ be another pattern not covered by the training set, but it satisfies the criteria of low HD with $\mathcal{P}$, then the pattern $\tilde{\mathcal{P}}$ will have high probability of lying in the same attractor basin, and consequently being predicted in the same class as that of $\mathcal{P}$.

MACA, as discussed earlier, acts as an implicit memory and consequently it can function as an effective pattern classifier. Genetic Algorithm (GA) formulation to arrive at the desired MACA realizing this specific objective of pattern classification has been proposed in [12] with O(n^3) complexity. The

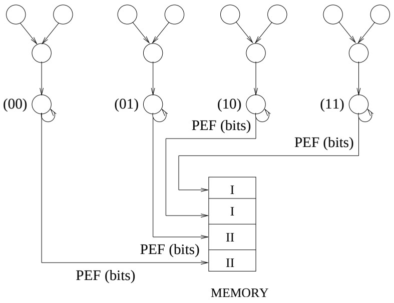


Figure 3. MACA based classification strategy

design reported in this paper achieves classification with linear complexity. It utilizes the concept of Dependency Vector/String introduced in next section. This formulation has enabled us to reduce the complexity of the classification algorithm from O(n^3) to O(n).

3. MACA Characterization Based on Dependency String

Identification of the PEF of the attractor basin, as discussed in *Section 2.3*, constitutes the main task for pattern classification. In [7, 8, 12, 24, 34, 35], an n-bit MACA has been characterized by its $n \times n$ Dependency Matrix (T) and its characteristic polynomial ($\phi(x)$). Generation of an attractor state $\acute{\mathcal{P}}$ from any state $\mathcal{P}$ ($\acute{\mathcal{P}} = T^d \cdot \mathcal{P}$) involves O($n^3$) complexity, where d is defined as depth (*Result III*). Consequently, the identification of the PEF of the attractor of the basin in which any state $\mathcal{P}$ belongs, involves O(n^3) complexity. In order to ensure the scalability on very large datasets, linear complexity of algorithm is highly desirable. This motivates us to undertake new characterization of MACA with the help of some linear operators other than Dependency Matrix.

In this section, we introduce the concept of new linear operators termed as Dependency Vector (DV) and Dependency String (DS), to characterize the attractor basins of an MACA. Applications of such operators bring down the complexity from O(n^3) to O(n) to identify the PEF of an attractor basin.

3.1. Dependency Vector (DV) and Dependency String (DS)

Consider a subspace with n-bit vectors of dimension ($n - m$). Then the cardinality of the subspace is $2^{(n-m)}$. If the set of these n-bit vectors is conceived as a system of n variable equations, then there will be m number of linear dependency relations as illustrated in the following example.

Example 3.1. Let the n (=5) bit vector subspace V be $\{$*00000, 01001, 10001,11000, 00110, 01111, 10111,11110*$\}$ with dimension 3 and m = 5 - 3 = 2. Now, V contains zero vector ($<00000>$) as an element. If the vector set (V) is conceived as a system of linear equations with five variables (a, b, c, d, e), then the elements of the vector subspace (V) can be rewritten in the form noted in Table 2.

In the set of equations of Table 2, $m = 2$; there are two linear dependency relations for all the vectors $v_i \in V$:

- $a \oplus b \oplus e = 0$; and
- $c \oplus d = 0$.

Table 2. System of Linear Equations representing a set of Vectors (V)

Vector (v_i)	Corresponding Linear Equation
00000	$0 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 0 \cdot e = 0$
01001	$0 \cdot a + 1 \cdot b + 0 \cdot c + 0 \cdot d + 1 \cdot e = 0$
10001	$1 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 1 \cdot e = 0$
11000	$1 \cdot a + 1 \cdot b + 0 \cdot c + 0 \cdot d + 0 \cdot e = 0$
00110	$0 \cdot a + 0 \cdot b + 1 \cdot c + 1 \cdot d + 0 \cdot e = 0$
01111	$0 \cdot a + 1 \cdot b + 1 \cdot c + 1 \cdot d + 1 \cdot e = 0$
10111	$1 \cdot a + 0 \cdot b + 1 \cdot c + 1 \cdot d + 1 \cdot e = 0$
11110	$1 \cdot a + 1 \cdot b + 1 \cdot c + 1 \cdot d + 0 \cdot e = 0$

In the context of above illustrative example (Table 2) we formally introduce the terms Dependency Vector and Dependency String.

Definition 3.1. Dependency Vector (DV) represents each individual linear dependency relationship of the variables supported by all the elements in the vector subspace (V). The two Dependency Vectors (DVs) for the illustrative *Example 3.1* are $< 11001 >$ and $< 00110 >$. Each of the n-bits signify the variables in that order - that is, the bits in the DV represents the variable in the sequence $< abcde >$. The 1's in the DV specify the dependent variables. In *Example 3.1*, while a, b and e are the dependent variables in first DV, c and d are the dependent variables in second DV. The modulo-2 sum (XOR) of the corresponding variables in all $v_i \in V$ = 0, as V contains zero vector ($< 00000 >$).

Definition 3.2. Dependency String (DS) represents the multiple linear dependency of the variables in the vector subspace (V). The Dependency String (DS) in the *Example 3.1* is $[11221]$ where '1' indicates the relationship between a, b and e while '2' indicates the relationship between c and d. In essence, the two Dependency Vectors (DVs) are merged together to form the Dependency String (DS).

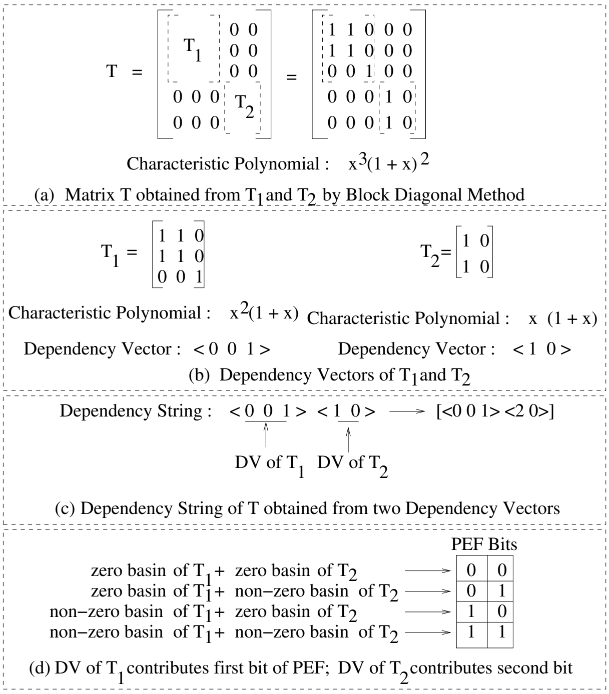
The 0-basin (with all zero vector as attractor, also referred to as zero basin) plays a significant role in characterizing an MACA [8]. As per the *Theorem 2.2*, the zero basin of an MACA having 2-attractor basins is a subspace of dimension $(n - 1)$. So there is one linear dependency relationship in the vector subspace and thus the characterization of the zero basin can be solely done by a Dependency Vector. Whereas the zero basin of an MACA with 2^m attractor basins is a subspace of dimension $(n - m)$. Therefore, there are m number of linear dependency relationships in the subspace. A Dependency String (DS) representing m number of DVs is necessary for characterization of such MACA.

Theorem 3.1. The Dependency String (DS) of an MACA with more than 2-attractor basins can be derived from the Dependency Vector (DV) of the MACA with 2-attractor basins.

Proof:
The Dependency Matrix (T) corresponding to the characteristic polynomial ($x^{n-m} \cdot (1 + x)^m$) can be obtained from m number of Dependency Matrices T_is ($i = 1, 2, \cdots, m$), where each T_i represents a 2-attractor MACA. The entire vector space produced by zero basin of an MACA with multiple attractor basins is the direct sum of individual vector spaces produced by each 2-attractor MACA [13].

The characteristic polynomial ($x^{n-m} \cdot (1+x)^m$) can be broken into m groups each containing one $(1+x)$, whereby we can claim that zero basin is formed as a result of direct sum of vector subspaces produced by m distinct groups. Each individual group has ($x^{n_i-1} \cdot (1+x)$) as its characteristic polynomial and so represents an MACA with 2-attractor basins.

So, the Dependency String (DS) of an MACA with 2^m-attractor basins can be derived from m number of Dependency Vectors (DVs) produced by each member of m groups. Hence the result follows. □



T =
T1
T2
Characteristic Polynomial : x3(1 + x) 2
(a) Matrix T obtained from T1 and T2 by Block Diagonal Method
T1 =
T2 =
Characteristic Polynomial : x2(1 + x)
Characteristic Polynomial : x (1 + x)
Dependency Vector : < 0 0 1 >
Dependency Vector : < 1 0 >
(b) Dependency Vectors of T1 and T2
Dependency String : < 0 0 1 > < 1 0 > [<0 0 1> <2 0>]
DV of T1
DV of T2
(c) Dependency String of T obtained from two Dependency Vectors
PEF Bits
zero basin of T1 + zero basin of T2
zero basin of T1 + non−zero basin of T2
non−zero basin of T1 + zero basin of T2
non−zero basin of T1 + non−zero basin of T2
(d) DV of T1 contributes first bit of PEF; DV of T2 contributes second bit

$$T = \begin{bmatrix} T_1 & \begin{matrix}0&0\\0&0\\0&0\end{matrix} \\ \begin{matrix}0&0&0\\0&0&0\end{matrix} & T_2 \end{bmatrix} = \begin{bmatrix} 1&1&0&0&0\\1&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&1&0 \end{bmatrix}$$

$$T_1 = \begin{bmatrix}1&1&0\\1&1&0\\0&0&1\end{bmatrix} \qquad T_2 = \begin{bmatrix}1&0\\1&0\end{bmatrix}$$

	PEF Bits	
zero basin of T1 + zero basin of T2	0	0
zero basin of T1 + non−zero basin of T2	0	1
non−zero basin of T1 + zero basin of T2	1	0
non−zero basin of T1 + non−zero basin of T2	1	1

Figure 4. Dependency String (DS) of Dependency Matrix (T) of Fig 1 which is formed through concatenation of two Dependency Vectors.
Note : (i) While the first Dependency Vector contributes the value of first pseudo-exhaustive bit, second Dependency Vector gives the value of another bit. (ii) Dependency String of T Matrix is obtained by concatenating two Dependency Vectors of T_1 and T_2 respectively - which are placed in non-overlapping positions.

Example 3.2. Fig 4 represents the Dependency String (DS) of the Dependency Matrix (T) of Fig 1 which is formed through concatenation of two Dependency Vectors (DVs). The T matrix (Fig 4(a)) of Fig 1 with characteristic polynomial ($x^3 \cdot (1+x)^2$) can be obtained from two matrices (T_1 and T_2) by the Block Diagonal Method of Fig 2. Fig 4(b) represents two characteristic polynomials - ($x^2 \cdot (1+x)$) and ($x \cdot (1+x)$), of T_1 and T_2 respectively. The DV of the two groups are $< 001 >$ and $< 10 >$ respectively. Consequently, the DS ([00120]) of the corresponding MACA (Fig 1) can be easily accomplished through concatenation of each individual DV (Fig 4(c)).

Derivation of the bits in pseudo-exhaustive field (PEF) as noted in Fig 4(d), has been explained in the subsection that follows.

3.2. Dependency String (DS) for Identification of PEF of An Attractor Basin

The following formulation provides the application of Dependency String (DS) to identify the PEF of an attractor basin.

According to *Section 3.1* (*Definition 3.1*), if DV is an n-bit Dependency Vector of $(n-1)$ dimensional vector subspace and $\mathcal{P}$ is an n-bit pattern, then the modulo-2 sum (XOR) of the dependent variables of $\mathcal{P}$ (where DV contains 1's) is equal to zero if $\mathcal{P}$ belongs to zero basin; otherwise 1. That is,

$$DV \cdot \mathcal{P} = \begin{cases} 0, & \text{if } \mathcal{P} \in \text{zero basin} \\ 1, & \text{if } \mathcal{P} \in \text{non-zero basin} \end{cases}$$

For example, if $DV = < 10111 >$ and $\mathcal{P} =< 11001 >$, then $DV \cdot \mathcal{P} =< 10111 > \cdot < 11001 >= 0$; that is $\mathcal{P} \in$ zero basin. So, an MACA with characteristic polynomial $x^{n-1} \cdot (1+x)$ can be represented by an n-bit Dependency Vector.

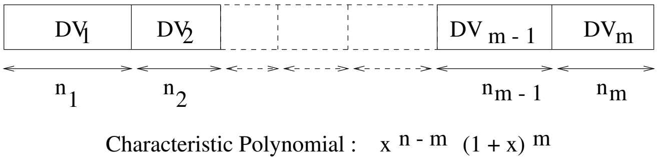


Figure 5. An n-bit Dependency String (DS) consists of m number of Dependency Vectors (DVs). Note: Each DV contributes the value of a pseudo-exhaustive bit (either 0 or 1) of an attractor basin.

A Dependency String (DS), consisting of m number of DVs - $DV_1, DV_2, \cdots, DV_m$, represents an MACA with characteristic polynomial ($x^{n-m} \cdot (1+x)^m$) (Fig 5). According to the *Theorem 2.1*, the number of pseudo-exhaustive bits will be m. Each DV contributes the value of a pseudo-exhaustive bit (either 0 or 1) of an attractor basin.

Let, an n-bit DS is produced through concatenation of m number of DVs of length $n_1, n_2, \cdots, n_m$ respectively (Fig 5), where $n_1 + n_2 + \cdots + n_m = n$ and $\mathcal{P}$ is an n-bit pattern whose attractor basin is to be identified. Then, for each DV_i (of length n_i), the dependent variables of the corresponding n_i bits of $\mathcal{P}$ (say $\mathcal{P}_i$) results in -

$$DV_i \cdot \mathcal{P}_i = \begin{cases} 0, & \text{if } \mathcal{P}_i \in \text{zero basin of } DV_i \\ 1, & \text{if } \mathcal{P}_i \in \text{non-zero basin of } DV_i \end{cases}$$

The resulting bit is the value of i^{th} pseudo-exhaustive bit. Finally, a string of m bits can be obtained from m number of DVs. This m-bit binary string is the pseudo-exhaustive field (PEF) of the attractor basins where the pattern $\mathcal{P}$ belongs.

Example 3.3. We illustrate the above concept with the example MACA of Fig 1. Suppose, we want to identify the attractor basin of a pattern $\mathcal{P} =< 10010 >$. We first synthesize a DS ($[001|20]$ of Fig 4) for which the distribution of patterns of MACA is similar to that of Fig 1. The DS of Fig 4 consists of two DVs ($< 001 >$ and $< 10 >$) of length 3 and 2 respectively. While the modulo-2 sum (XOR) of dependent variables of first 3 bits of $\mathcal{P}$ ($\mathcal{P}_1$) contributes the first pseudo-exhaustive bit as 0, rest 2 bits

($\mathcal{P}_2$) contribute 1. This is explained in Fig 4(d). So, the PEF of corresponding attractor basin containing $\mathcal{P}$ is 01.

In [7, 8, 12], when the MACA (T) of Fig 1 is loaded with the pattern $\mathcal{P} =< 10010 >$, it travels through a number of states equal to the depth ($d = 2$) of the MACA and ultimately reaches an attractor state $\acute{\mathcal{P}} =< 00011 >$ - that is,

$$\acute{\mathcal{P}} = T^d \cdot \mathcal{P} \tag{1}$$

where T is the $n \times n$ Dependency Matrix of Fig 1. The pseudo-exhaustive field (3rd and 4th bit, - that is PEF = 01) of the attractor basin is identified as per the algorithm reported in [8]. The complexity of the algorithm is O(n^3), n being the size of the pattern. Whereas in the proposed scheme, the pseudo-exhaustive field (PEF) of the attractor basin of $\mathcal{P}$ is given by

$$PEF = DS \cdot \mathcal{P} \tag{2}$$

where DS and $\mathcal{P}$ - both are n-bit vectors. So, the complexity of this approach is O(n). Thus, to identify the PEF of an attractor basin of a pattern, Dependency String can be employed rather than the Dependency Matrix (T) which reduces the complexity from O(n^3) to O(n).

In order to reduce the search space for identifying the desired MACA to be employed for pattern classification, we next introduce the concept of Valid Dependency Vector/String.

3.3. Valid Dependency Vector/String

A 3-neighborhood MACA whose next state depends on itself, its left neighbor and right neighbor, cannot produce all the variations of Dependency Vector/String. In the present context, the Dependency Vector/String which can be generated by this MACA is termed as Valid Dependency Vector/String respectively. The following theorems set the guideline for determination of Valid Dependency Vector/String.

Theorem 3.2. [11] The vector subspace of a 3-neighborhood n-cell MACA with two attractor basins cannot generate a Dependency Vector (DV) of the form $\cdots 1 \cdots (\ \acute{k}\ 0's\ ) \cdots 1 \cdots$ with $\acute{k}$ number of 0's between two 1's, where $\acute{k} \geq 2$.

Example 3.4. The examples of Dependency Vectors which cannot be generated by a 3-neighborhood MACA are $< 100011 >$, $< 1001001 >$. In all these cases, there are more than 1 zeroes between consecutive ones and hence these are termed as invalid. While some valid Dependency Vectors are $< 101011 >$, $< 00010111 >$ etc.

Theorem 3.3. A Valid Dependency String has the constituent Valid Dependency Vectors placed in non-overlapping positions.

Proof:
Let, the characteristic polynomial of a Dependency Matrix (T) is given by ($x^{n-m} \cdot (1 + x)^m$) for n-cell MACA with 2^m-attractor basins. According to the results of *Section 2.2*, the characteristic polynomial can be written as ($x^{d_1} \cdot (1 + x) \cdot x^{d_2} \cdot (1 + x) \cdots x^{d_m} \cdot (1 + x)$), where $d_1 + d_2 + \cdots + d_m = n - m$. Let $T_1, T_2, \cdots, T_m$ are the Dependency Matrices corresponding to the characteristic polynomials $x^{d_1} \cdot (1+x)$, $x^{d_2} \cdot (1 + x)$, $\cdots$, $x^{d_m} \cdot (1 + x)$ respectively. So, $T_1, T_2, \cdots, T_m$ are the MACA with 2-attractor basins.

Now, the T matrix with characteristic polynomial $(x^{n-m} \cdot (1+x)^m)$ can be obtained from all these matrices $(T_1, T_2, \cdots, T_m)$ by the Block Diagonal Method of Fig 2 where T_i (T_j) has no dependency on T_j (T_i). The entire vector subspace produced by zero basin of MACA (T) will be the direct sum of individual vector subspaces produced by the zero basin of each T_i's. Also, the Dependency Vector corresponding to T_i does not affect that of T_j and vise versa. As a result, the Dependency String of T matrix can be obtained through concatenation of m number of Dependency Vectors of $T_1, T_2, \cdots, T_m$ respectively - which are placed in non-overlapping positions (Fig 4). Hence, the result follows. □

Example 3.5. An example Valid Dependency String is $[101012020]$, whereas the following Dependency String - $[12010]$ - where 1 and 2 are interleaved is invalid; that is, the vectors of the zero basin of an MACA will not generate such invalid Dependency String.

Characterization of MACA based on Dependency Vector/String establishes that MACA acts as an implicit memory. It also acts as a natural classifier. In designing MACA based pattern classifier, in stead of storing the Dependency Matrix (T) of the MACA, we store only DV/DS. Naturally, memory overhead of the classifier goes down significantly.

4. Design of Multiple Attractor CA Based Two Stage Pattern Classifier

To enhance the classification accuracy of the machine, we have refined the approach reported in [12] and report a new CA based classifier. It can classify an n-bit pattern with O(n) complexity. Multi-class classifier is built by recursively employing the concept of two class classifier. The MACA based classifier proposed here has two distinct stages and hence referred to as Two Stage Classifier (Fig 6).

4.1. MACA Based Two Stage Classifier

The design of MACA based classifier for two n-bit pattern sets S_1 and S_2 should ensure that elements of one class (say S_1) are covered by a set of attractor basins that do not include any member from the class S_2 and vice versa. Consequently, any two n-bit patterns $\mathcal{P}_1 \in S_1$ and $\mathcal{P}_2 \in S_2$ should fall in different attractor basins.

The design of the proposed Two Stage Classifier has been elaborated in the subsequent discussions. Let, an MACA corresponding to the characteristic polynomial $(x^{n-m} \cdot (1+x)^m)$ can classify two n-bit pattern sets S_1 and S_2. That is,

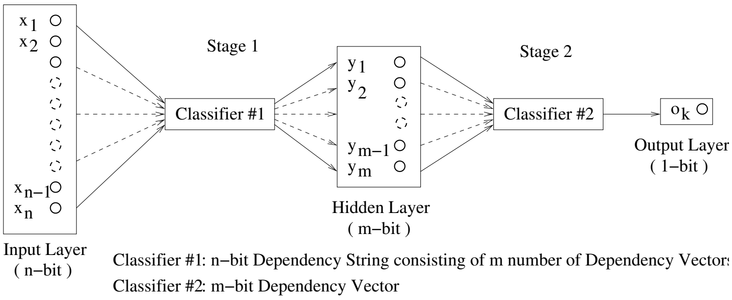
$$DS \cdot \mathcal{P}_1 \neq DS \cdot \mathcal{P}_2 \tag{3}$$

where DS is an n-bit Dependency String consisting of m number of Dependency Vectors (Fig 5). Then, the total number of attractor basins will be 2^m and the pseudo-exhaustive field (PEF) (*Theorem 2.1*) of each attractor basin will be an m-bit binary pattern/string. The Stage 1 of the proposed Two Stage Classifier maps n-bit patterns to m-bit patterns representing the PEF of attractor basins.

Next, for two class classification the m-bit patterns are classified into two distinct classes. This task is handled by Stage 2 of the Two Stage Classifier. Let, k_1 and k_2 be two m-bit pattern sets consisting of pseudo-exhaustive bits of attractors of two n-bit pattern sets S_1 and S_2 respectively. Then, k_1 and k_2 can also be regarded as two m-bit pattern sets for two class classification. So, we synthesize a 2-attractor basin MACA based two class classifier. While one class (say k_1) belongs to one attractor basin, another attractor basin houses the elements of the class k_2. Any two m-bit patterns $p_1 \in k_1$ and $p_2 \in k_2$ should fall in different attractor basins (while one in zero basin, another in non-zero basin), - that is,

$$DV \cdot p_1 \neq DV \cdot p_2 \tag{4}$$

where DV is an m-bit Dependency Vector.



Classifier #1: n−bit Dependency String consisting of m number of Dependency Vectors
Classifier #2: m−bit Dependency Vector

Figure 6. Two Stage Classifier

Fig 6 represents the architecture of Two Stage Classifier. It consists of three layers - input, hidden and output layers denoted as x_i $(i = 1, 2, \cdots, n)$, y_j $(j = 1, 2, \cdots, m)$, and o_k $(k = 1)$ respectively. While the first classifier (Classifier #1) maps an n-bit pattern of the input layer into an m-bit pattern (PEF) of the hidden layer, the second classifier (Classifier #2) maps that m-bit pattern into a single bit (either 0 or 1) of the output layer. That is, Classifier #1 provides an appropriate mappings of patterns of input layer into PEF (pseudo-exhaustive field) of the hidden layer and the Classifier #2 implements the classification of the PEFs rather than the original patterns.

Let, x be an n-bit input pattern whose class is to be identified by MACA based Two Stage Classifier. At first, x is loaded with the Classifier #1 which outputs an m-bit pattern y (pseudo-exhaustive field of attractor of the basin where x belongs). That is,

$$y = DS \cdot x \tag{5}$$

Next, y is loaded with the Classifier #2 which gives a single value o (pseudo-exhaustive field of attractor of the basin where y belongs) that determines the class of the input pattern x. That is,

$$o = DV \cdot y \tag{6}$$

In order to evolve the MACA based Two Stage Classifier (two MACAs for Stage 1 and Stage 2 respectively) realizing this design objective, we have developed a special type of Genetic Algorithm (GA) formulation.

4.2. Genetic Algorithm (GA) for Evolution of Two Stage Classifier

The basic structure of GA [18] revolves around the concept of encoding a solution in bit string format referred to as chromosome and evolving the successive solutions according to its fitness. The three major functions of GA - Random Generation of Initial Population (IP), Crossover and Mutation, as developed in the current GA formulation for evolution of Two Stage Classifier, are next discussed.

4.2.1. Chromosome

Rather than the conventional bit string, the proposed scheme employs a chromosome which consists of two parts:

- a Dependency String (DS) for Classifier #1 - a symbol string of numerical digits; and
- a Dependency Vector (DV) for Classifier #2 - a binary string with valid format defined in *Theorem 3.2* and *Theorem 3.3.*

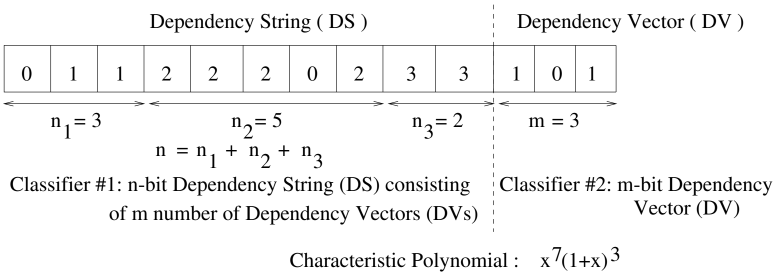

Dependency String (DS)
Dependency Vector (DV)
0 1 1 2 2 2 0 2 3 3 1 0 1
$n_1 = 3$
$n_2 = 5$
$n_3 = 2$
m = 3
$n = n_1 + n_2 + n_3$
Classifier #1: n-bit Dependency String (DS) consisting of m number of Dependency Vectors (DVs)
Classifier #2: m-bit Dependency Vector (DV)
Characteristic Polynomial : $x^7(1+x)^3$

Figure 7. An example chromosome for current GA formulation

So, the length of a chromosome is equal to $(n+m)$ where n is the number of bits in a pattern and m is the number of pseudo-exhaustive bits. Fig 7 represents a chromosome corresponding to the characteristic polynomial $(x^7 \cdot (1+x)^3)$. It consists of a 10-bit $(= n)$ DS $[011|22202|33]$ (Classifier #1) and a 3-bit $(= m)$ DV $< 101 >$ (Classifier #2). The Dependency String (DS) has partition points at 3rd and 8th positions and the corresponding DVs are of length 3 $(= n_1)$, 5 $(= n_2)$ and 2 $(= n_3)$ respectively.

4.2.2. Random Generation of Initial Population

To form the initial population, it must be ensured that each solution randomly generated is a combination of an n-bit DS with 2^m number of attractor basins (Classifier #1) and an m-bit DV (Classifier #2). The chromosomes are randomly synthesized according to the following steps.

1. Randomly partition n into m number of integers such that $n_1 + n_2 + \cdots + n_m = n$.
2. For each n_i, randomly generate a valid Dependency Vector (DV).
3. Synthesize Dependency String (DS) through concatenation of m number of DVs for Classifier #1.
4. Randomly synthesize an m-bit Dependency Vector (DV) for Classifier #2.
5. Synthesize a chromosome through concatenation of Classifier #1 and Classifier #2.

Fig 7 represents a randomly generated 13-bit chromosome which is produced through concatenation of 10-bit DS (Classifier #1) and 3-bit DV (Classifier #2), while the 10-bit DS is produced through concatenation of three DVs of length 3, 5 and 2 respectively.

4.2.3. Crossover Algorithm

The crossover algorithm implemented is similar in nature to the conventional one normally used for GA framework with minor modifications as illustrated below. The algorithm takes two chromosomes from the present population (PP) and forms the resultant chromosome. Like a single point crossover, it sets a crossover point and each half about the crossover point is selected from the two respective chromosomes.

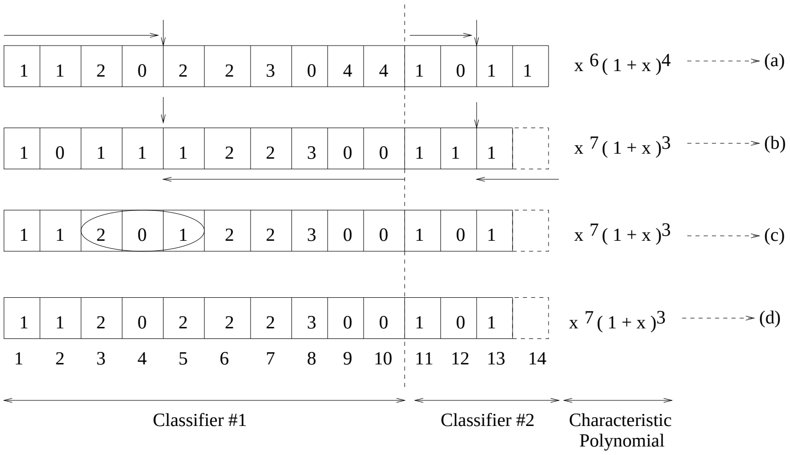


Figure 8. An example of crossover technique

Fig 8 shows an example of the crossover process. Two chromosomes with characteristic polynomials ($x^6 \cdot (1 + x)^4$) and ($x^7 \cdot (1 + x)^3$) respectively are shown in Fig 8(a) and Fig 8(b). The single point crossover is performed in two parts separately. The crossover points are selected randomly which are 4 in first part (Classifier #1) and 12 in second part (Classifier #2). So, the first 4 symbols are taken from first chromosome and the next 6 symbols are taken from second chromosome to form Classifier #1. Similarly, 11th and 12th bits are taken from first chromosome and 13th bit is taken from second chromosome to form Classifier #2. But due to this crossover, the resulting chromosome, as explained below, generates an invalid DS due to the symbols in 3rd, 4th and 5th positions (encircled in Fig 8(c)).

In Fig 8(c), the DS (Classifier #1) is $[1120122300]$ where 1 and 2 are interleaved, which is invalid (*Theorem 3.3*). The resultant valid chromosome after local recoding of symbols is shown in Fig 8(d).

4.2.4. Mutation Algorithm

The mutation algorithm emulates the conventional mutation scheme. It makes some minimal change in the existing chromosome of PP (Present Population) to form a new chromosome for NP (Next Population). Similar to conventional single point mutation, the chromosome is mutated at a single point as illustrated in Fig 9. The mutation points for Classifier #1 and #2 are 4 and 12 respectively.

Any anomaly in respect of validity of Dependency Vector/String, if appears, is resolved to ensure that the mutated chromosome is also a valid chromosome. The inconsistent format, as shown in Fig 9(b) is the mutated version of Fig 9(a). The inconsistency of the chromosome of Fig 9(b) is resolved through local recoding of symbols to generate the valid format of Fig 9(c).

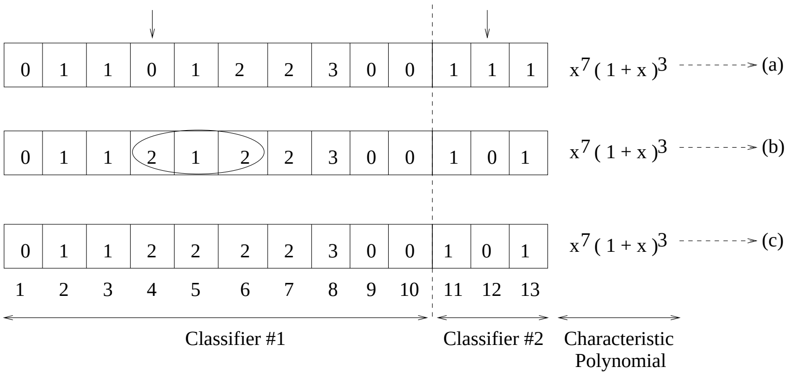


Figure 9. An example of mutation technique

4.2.5. Fitness Function

In classification, the fitness F of a particular chromosome in a population is determined by two factors.

- The capability of the evolved DS (Classifier #1) for classifying the given input pattern set S_1 and S_2 into separate set of attractor basins - this is referred to as the factor F_1.
- The capability of the evolved DV (Classifier #2) for classifying the pseudo-exhaustive field (PEF) set k_1 and k_2 into different attractor basins (while one in zero basin, another in non-zero basin) - this is referred to as the factor F_2.

The fitness function F of a particular chromosome is given by

$$F = F_1 \cdot F_2 \tag{7}$$

The experimental results reported in *Sections 5, 6* and *7*, confirm that this relation, although evolved empirically, provides the desired direction of GA evolution to arrive at the best solution for classification. Next subsection reports the performance of the proposed Two Stage Classifier.

4.3. Performance Analysis

To evaluate the efficiency of the proposed pattern classifier, we perform extensive experiments for different values of n (number of bits in a pattern) and t (size of dataset - that is, number of tuples in the dataset). The pattern sets are generated according to the method proposed in [12]. The Genetic Algorithm (GA) has been evolved for maximum 100 generations. All the experiments are performed in SUN with Solaris 5.6, 350 MHz clock.

Table 3 reports the efficiency of the Two Stage Classifier. Column I and II of Table 3 represent different topologies (n : m : 1; where n and m are the number of bits in the input layer and hidden layer respectively) and size of the datasets respectively. Column III and IV depict the classification accuracy and memory overhead of the proposed pattern classifier. The classification accuracy is defined as the percentage of test data that can be correctly classified. The memory overhead of the proposed classifier is the memory required to store the n-bit DS (Classifier #1) and m-bit DV (Classifier #2). So, the number

Table 3. Efficiency of MACA based Two Stage Classifier

Topology ($n : m : 1$)	Size of Dataset	Classification Accuracy (%)	Memory Overhead	Generation Time (ms)	Retrieval Time (ms)
100:5:1	5000	97.13	105	454	161
	10000	97.41	105	461	298
200:7:1	5000	98.03	207	1102	301
	10000	97.83	207	1139	578
300:7:1	5000	98.03	307	1573	438
	10000	97.83	307	1581	849
400:10:1	5000	96.93	410	1789	606
	10000	97.03	410	1809	1125
500:10:1	5000	96.43	510	2344	710
	10000	96.71	510	2339	1403

of bits required to store each Two Stage Classifier is equal to $(n + m)$. In Column V and VI we provide the generation and retrieval time of the proposed pattern classifier. The following conclusions can be derived from this experimental results.

- The memory overhead of Two Stage Classifier, as per Column IV, is independent of the size of datasets. The low memory overhead, independent of dataset size, has been achieved due to the fact that -

    1. MACA functioning as pattern classifier acts as an implicit memory; and
    2. MACAs are stored by their DV/DS.

- The generation and retrieval time, as the results of Column V and VI indicate, are linear in nature.

High classification accuracy, low memory overhead, and linear time complexity of the proposed pattern classifier have been validated through extensive experimentations. Next three sections report the pattern classification results in three different application domains - data mining, image compression, and fault diagnosis.

5. Data Mining

Data Mining/Knowledge Discovery in Databases (KDD) [21, 26] can be defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from a database [25]. One of the important problems in data mining is classification [2]. Classification based data mining has been proposed in the fields of medical diagnosis, performance prediction, selective marketing, etc. Solutions based on Bayesian Classification [9], Neural Networks [17, 20], Genetic Algorithms [14], Decision Trees [22, 28, 32], etc., have been reported. However, in all these cases [9, 22, 28, 32], the algorithms require a data structure proportional to the number of tuples to stay memory resident. This restriction puts a hard limit on the amount of data that can be handled by these classifiers. In other words, the issue of scalability is the source of major concern.

In the above background, the Two Stage Classifier proposed in this paper achieves following:

- it refines the scheme proposed in [12] by enhancing classification accuracy, minimizing memory overhead; and
- linear complexity of classification in prediction phase.

5.1. Experimental Setup

To analyze the performance of Two Stage Classifier, we perform extensive experiments on the datasets available from http://www.ics.uci.edu/∼mlearn/MLRepository.html. The classification accuracy and the memory overhead of the Two Stage Classifier are compared with different standard classification algorithms such as Bayesian [9], C4.5 [28], MLP (Multilayer Perceptron) [17], First Generation MACA [12], etc. To handle real data having categorical and/or continuous attributes, the dataset is suitably modified using Data Discretization [16] and Thermometer Coding [10] to fit the input characteristic of the proposed classifier. All the experiments are performed in SUN with Solaris 5.6, 350 MHz clock.

5.2. Classification Accuracy

Table 4 compares the classification accuracy of Two Stage Classifier with that of different standard classification algorithms. The experimental results of Table 4 clearly establish that the classification accuracy of proposed Two Stage Classifier is comparable to that of different standard classification algorithms [9, 12, 17, 28].

Table 4. Classification Accuracy of Different Algorithms in Data Mining Application

Dataset	Bayesian	C4.5	MLP	First Generation MACA [12]	Two Stage Classifier
monk1	99.9	100	100	100	100
monk2	69.4	66.2	75.16	76.24	78.16
monk3	92.12	96.3	76.58	97.01	97.17
crx	83.14	84.5	74.29	83.87	86.12
labor-neg	83.07	82.4	89.03	87.37	89.14
vote	92.37	94.8	90.87	95.06	95.88
hypo	98.18	99.4	94.13	99.51	99.59
Australian	83.4	85.8	84.7	86.42	86.47
Diabetes	72.9	74.2	75.3	75.93	75.93
DNA	90.3	93.3	91.4	87.99	87.96
German	66.79	67.4	67.12	74.62	74.64
Heart	80.12	79.3	80.74	86.24	86.59
Satimage	85.4	85.2	86.2	77.45	77.49
Shuttle	99.9	99.9	99.6	94.09	94.03
Letter	87.4	86.6	67.2	84.13	84.41
Vehicle	72.9	68.5	79.3	76.2	78.7
Segment	96.7	94.6	94.4	89.4	89.5

Table 5. Memory Overhead of Different Algorithms in Data Mining Application

Dataset	Bayesian	C4.5	MLP	First Generation MACA [12]	Two Stage Classifier
monk1	1.7	2.77	0.35	0.08	0.02
monk2	2.3	3.95	0.35	0.08	0.02
monk3	1.7	2.38	0.36	0.08	0.02
crx	22	24.35	2.25	65.94	0.15
labor-neg	2.4	3.44	2.56	2097.59	0.17
vote	13	14.35	2.66	0.21	0.06
hypo	213	214.54	6.32	132.75	0.62
Australian	19	37.85	1.93	0.42	0.13
Diabetes	14	27.15	0.44	0.62	0.13
DNA	1000	1067.96	37.22	3.13	0.38
German	49	99.31	13.64	0.71	0.16
Heart	9.8	19.46	1.52	0.38	0.12
Satimage	669	709.72	11.33	171.96	2.78
Shuttle	1500	1513.57	0.71	3.22	0.85
Letter	766	1299.28	2.57	43.67	6.25
Vehicle	47.02	72.14	1.32	0.39	0.10
Segment	121.89	370.42	2.74	0.52	0.23

5.3. Memory Overhead

The memory overhead of the proposed MACA based Two Stage Classifier, as discussed earlier, is equal to $(n + m)$, where n is the number of bits in the input layer and m is the number of bits in the hidden layer (Fig 6). Consequently, the total memory overhead to design MACA based multi-class classifier with K number of classes is

$$MO = (K - 1) \cdot (n + m) \tag{8}$$

Table 5 reports the comparison of memory overhead of different classification algorithms in terms of KByte. The results clearly establish that the memory overhead to implement the proposed Two Stage Classifier is significantly lesser compared to that of other classification algorithms [9, 12, 17, 28]. Also, the memory overhead of the proposed classifier is independent of the size of datasets.

6. Image Compression

Lossy data compression is a process of reducing the amount of data required to represent a given quantity of information with acceptable loss. It removes redundancy, repeatability and irrelevancy of data blocks of input file to generate the compressed output. In order to demonstrate the capability of Two Stage Classifier in compression technology, we have concentrated on lossy compression for human portrait. The details of this research has been reported in [33]. In this paper we highlight the design of Two Stage Classifier for this specific application.

The well known Vector Quantization (VQ) method has been applied to generate the codebook from the training set of human portrait [15]. The MACA has been used as an implicit memory to store the codebook and search for the best match of an input data block. The encoding time is reduced substantially by employing the MACA based Two Stage Classifier reported in this paper. The excellent compression ratio with acceptable image quality, establishes the efficiency of Two Stage Classifier for this application.

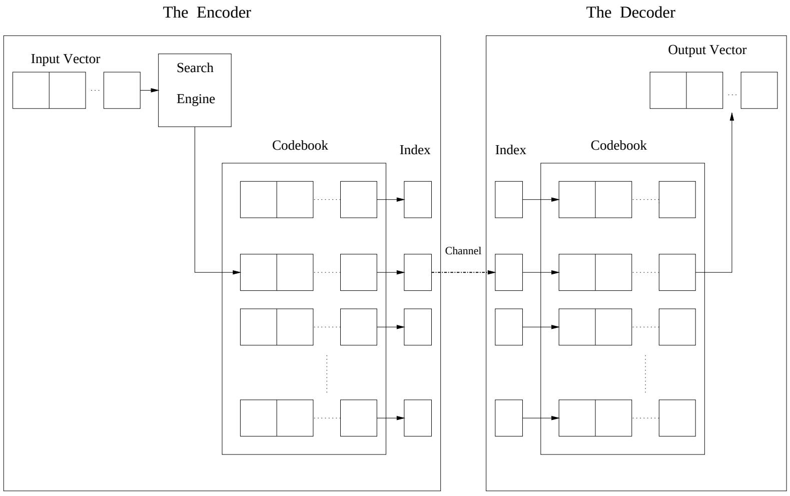


Figure 10. Encoder and Decoder

Vector Quantization (VQ) is a lossy data compression method [15]. It maps the n-dimensional vectors in the vector space R^n into a finite set of vectors stored in a codebook. Each vector of the codebook is known as codeword. A cluster is the set of vectors having minimum deviation from a specific codeword in the codebook. Thus each codeword (also known as codevector) is the nearest neighbor of the set of vectors in a cluster. A VQ method mainly consists of two operations:

- an encoder - to encode each block of input image file with the index of a codeword in the codebook;
- a decoder - to get back the representative block from the codebook.

The encoder, as shown in Fig 10, takes an input vector and outputs the index of the corresponding codeword from the codebook that gives minimum deviation. The index of the codeword is sent to the receiver end. The decoder, on receiving this index file, replaces each entry with the associated codeword found from the codebook kept on the receiver side. Codebook generation plays a key role in VQ scheme.

6.1. Codebook Design

Codebook design consists of two steps: design of training set and generation of codebook. Each step is illustrated with reference to Fig 11.

6.1.1. Design of training set

The training set has been designed out of 20 different human-face images with wide variation of pixel values. Each image of training set is segmented into 16×16 blocks that is subsequently processed in following three sequential steps:

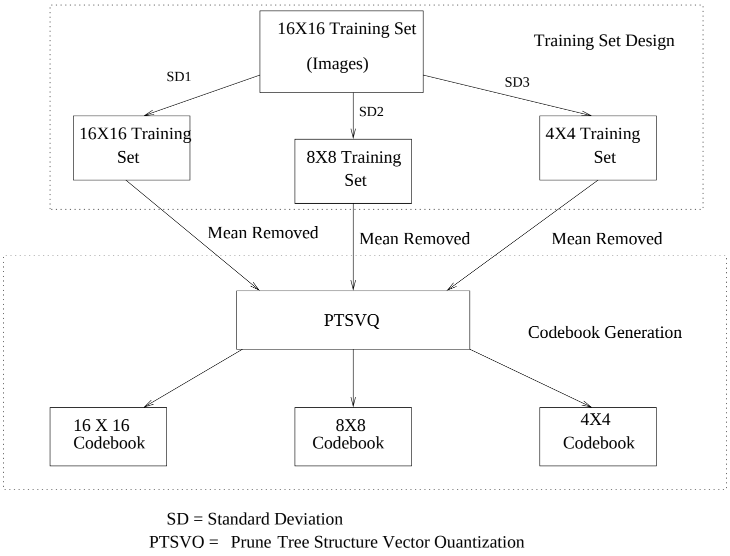


Figure 11. Block diagram of codebook generation scheme

- Step 1: Calculate the norm (standard deviation) of each 16×16 pixel block. If the norm matches with the pre-specified criteria (say SD1 as shown in Fig 11), it is stored as a member of 16×16 training set, else referred to as 16×16 residual training set to be processed in Step 2.
- Step 2: Each member of the residual 16×16 training set is broken into four 8×8 pixel blocks. Next, we calculate the norm of each 8×8 block and compare with the pre-specified criteria (say SD2 as shown in Fig 11). If the norm matches with the criteria, then it is stored as 8×8 training set. Otherwise it is referred to as the residual of 8×8 training set to be processed in the next step.
- Step 3: Each 8×8 residual block is broken into four 4×4 pixel blocks. Calculate the norm of each 4×4 block and compare with the pre-specified criteria (say SD3 as shown in Fig 11). If the norm matches with the criteria, then it is stored as 4×4 training set. Otherwise the block is broken into four 2×2 pixel blocks and treated as 2×2 training set.

The matching criteria SD (standard deviation) has been fixed on the basis of statistical characteristics of 16×16, 8×8 and 4×4 pixel blocks of training set [33].

6.1.2. Codebook Generation

To design the codebook for three training sets (16×16, 8×8 and 4×4) we have used Prune Tree Structured Vector Quantization (PTSVQ) [15] method. Three codebooks are generated from the three different training sets as shown in Fig 11. The mean value is computed for each training set. The PTSVQ is applied on the mean removed vectors. Each element of the vector after subtraction of the mean value is known as mean removed vector. At the time of encoding, a 16×16 pixel block is sequentially taken from the image and depending on the match criteria it is coded either by the codebook indices for 16×16 or broken to four 8×8 blocks and encoded with the indices of 8×8 codebook. For a 8×8 pixel block, if a proper match in the codebook is absent, it is treated as a collection of four 4×4 blocks and coded by four indices from the 4×4 codebook. A separate match file is kept to track the sequence of indices from different codebooks.

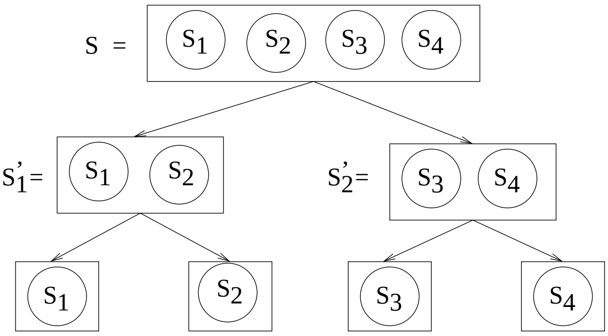


Figure 12. Logical structure of multi-class classifier equivalent to PTSVQ

In binary PTSVQ, if K is the total number of codebook entries, the depth of binary tree is $\log_2 K$. So, the time required to search the best match for an input block is $\log_2 K$ comparisons of $x \times x$ $(x = 4, 8, 16)$ pixel block size. Reduction of this comparison is necessary for on line transmission of image data. The Two Stage Classifier proposed in *Section 4*, is used to reduce this comparison and search time for best match in the codebook.

The binary search for best match in the codebook is implemented with an MACA based multi-class classifier realized with multi-stage two class classifiers. In order to identify the best match in binary PTSVQ scheme, the input vector is compared with two centroids of two vector clusters in each layer of the tree and one of the branches, as shown in Fig 12, is selected according to matching criterion. A sequence of comparisons are done in subsequent levels till the leaf node is reached. We have designed MACA based Two Stage Classifier to model the comparison operation at each node of PTSVQ binary tree. The pixel blocks of the training sets employed for design of codebook and PTSVQ binary tree are also used as input for design of Two Stage Classifier. A set of Two Stage Classifiers are generated that acts as multi-class classifier of the vectors in a codebook.

Fig 12 illustrates the design of MACA set for a codebook. Suppose, we want to classify the pattern set $S = \{\{S_1\}, \{S_2\}, \{S_3\}, \{S_4\}\}$ into four classes such that the classifier would output the correct class i $(i = 1, 2, 3, 4)$ for a given input vector $\mathcal{P}_i \in \{S_i\}$. At the first level, we divide whole pattern set S into two classes - $\acute{S}_1$ and $\acute{S}_2$, where $\acute{S}_1 = \{\{S_1\}, \{S_2\}\}$ and $\acute{S}_2 = \{\{S_3\}, \{S_4\}\}$. The Two Stage Classifier is designed to classify two distinct classes $\acute{S}_1$ and $\acute{S}_2$. Fig 12 represents two classes $\acute{S}_1$ and $\acute{S}_2$. The same process is then applied for $\acute{S}_1$ and $\acute{S}_2$ to isolate $\{S_1\}$, $\{S_2\}$ and $\{S_3\}$, $\{S_4\}$ respectively and to generate two Two Stage Classifiers. Thus the logical structure of the multi-class classifier is equivalent to the PTSVQ binary tree representing a codebook.

For a given vector $\mathcal{P}_1$ $(\mathcal{P}_1 \in S_1)$, we need to identify the codebook entry (that is codeword) closest to $\mathcal{P}_1$. At the prediction phase, the vector $\mathcal{P}_1$ is given as input and its class is identified as follows. At the first stage, the classifier designed with the Two Stage Classifier is loaded with $\mathcal{P}_1$ and allowed to run. It returns the desired class $\acute{S}_1$. In the next level, the classifier is loaded with $\mathcal{P}_1$ to output the class S_1.

6.2. Experimental Results of CA based Compression

The algorithm is applied on different standard pictures of human face. The experimental results are reported in Table 6. Table 6 represents the PSNR values as well as compression ratio of the set of images when compressed and decompressed using the proposed scheme. Fig 13 show the comparative study of original images and the decompressed images. The experimental results of Fig 13 and Table 6 confirm high PSNR value with a compression in the range of 95% to 97.5%.

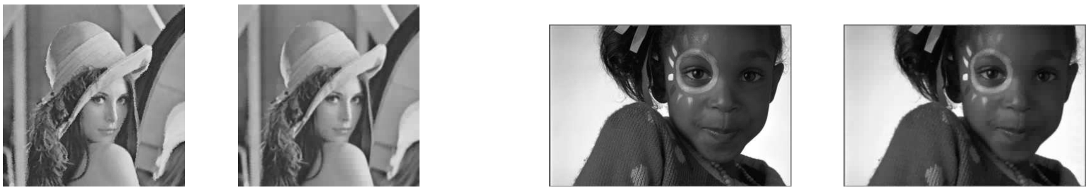


Figure 13. Original and decompressed image of (i) lena with compression 96.43% and PSNR 32.81; (ii) girl with compression 95.66% and PSNR 34.27

Table 6. Results on Static Image

Image File	PSNR	Compression (%)
lena	32.81	96.43
girl	34.27	95.66
Proj100	35.02	96.54
Proj129	37.88	97.63
Proj131	34.01	95.22
Proj138	33.69	97.55
Proj140	30.02	96.68
Proj146	37.55	97.57
Proj148	38.06	97.75
Proj192	30.06	97.17
Proj210	34.24	96.41

Table 7. Execution Time (in milli seconds)

Block Size	Full Search	PTSVQ	MACA
4×4	0.0121	0.00824	0.00562
8×8	0.0473	0.03312	0.01367
16×16	0.1941	0.13192	0.04102

Fig 14 depicts the rate distortion behavior of the proposed scheme and other standards [31]. Rate distortion means the change of PSNR at different bit representation of each pixel. It is seen that, the proposed scheme for specific domain outperforms all other algorithms - the JPEG 2000 reversible (J2K-R),

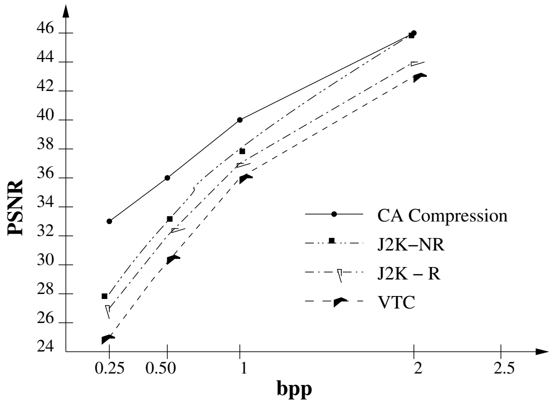


Figure 14. PSNR of all test images for CA based compression and other standard algorithms when performing lossy encoding at 0.25, 0.5, 1 and 2 bpp and other standard

non-reversible (J2K-NR), JPEG (older technology) and MPEG-4 VTC at any given bit rate. The size of test images are 512×512 and depth 8 bits per pixel.

Table 7 represents the comparison of execution time of the proposed MACA based Two Stage Classifier and that of the full search and PTSVQ. While Column I of Table 7 depicts the size of the block, Column II, III, and IV report the execution time taken by a block to search its closest codevector in a tree of height 14. The number of codevector is 2^{14}. All the results reported in Table 7 confirm that the execution time of the proposed scheme is significantly small compared to full search and PTSVQ.

All the results reported above confirm the following facts:

- The excellent compression ratio with high PSNR establishes the high classification accuracy of the Two Stage Classifier.
- MACA based Two Stage Classifier acts as an implicit memory to store the codebook and search for the best match.
- The encoding time is reduced substantially by employing proposed pattern classifier.

Next section reports the application of Two Stage Classifier in fault diagnosis of electronic circuits.

7. Fault Diagnosis of Electronic Circuits

This section formulates fault diagnosis in electronic circuits as a pattern classification problem. The proposed pattern classification scheme has been projected as a classifier of faulty response-patterns of a circuit leading to diagnosis of faulty module. The Genetic Algorithm (GA) (*Section 4.2*) is employed to synthesize the desired MACA based Two Stage Classifier required for diagnosis of a CUT (Circuit Under Test). The CUT is assumed to have a network of large number of circuit components partitioned into a number of sub-circuits referred to as modules. Introduction of GA significantly reduces the design overhead of the MACA based classifier that supports:

- low memory overhead for diagnosis - reduction of one to two order of magnitude of memory overhead has been achieved over that required for fault dictionary based diagnosis scheme;

- excellent diagnostic resolution;
- low diagnostic aliasing; and
- low cost hardware implementation of a generic Fault Diagnosis Machine (FDM) with simple, regular, modular, and cascadable structure of CA that suits ideally for VLSI implementation.

The FDM can be viewed as a Watch-Dog processor intended for high speed on-line diagnosis for critical application areas. The following scenario motivated us to undertake this research.

In order to improve the product quality, reduce time to market, and cut down production cost, the demand for fault diagnosis in an electronic circuit has greatly increased. The objective of fault diagnosis is to guide the test engineers to search the physical location of the defect on a circuit in the early production phase. Use of fault dictionaries is a probable solution for the diagnosis process, particularly when repeated diagnosis is required for different copies of the same circuit [1, 3, 4]. But this scheme becomes inefficient for a sufficiently large circuit due to the large volume of fault dictionary. Different schemes to compact the size of the dictionary have been proposed [5, 27, 30]. However, such compaction reduces diagnostic resolution since multiple response patterns for different faults get compressed to the identical signature. The best test environment for diagnostic purposes should differentiate between all faults that are distinguishable. In the above background, the diagnosis scheme employs Two Stage Classifier (*Section 4*) which effectively provides an implicit storage mechanism of voluminous response data. Consequently it provides the solution to the problem of space and time associated with handling large fault dictionary. The details of this MACA based fault diagnosis scheme has been reported in [24, 34, 35]. In this paper we highlight the impact of introduction of the proposed MACA based Two Stage Classifier for efficient diagnosis of electronic circuits.

7.1. Circuit Diagnosis - A Pattern Classification Problem

A circuit has been assumed to consist of a number of sub-circuits or modules. The diagnosis framework has been designed to diagnose the faulty module in the circuit. Let, for a circuit with K number of modules - $M_1, M_2, \cdots, M_K$, $\{S_{11}, S_{12}, \cdots\}$, $\{S_{21}, S_{22}, \cdots\}$, $\cdots$, $\{S_{K1}, S_{K2}, \cdots\}$ be the signature sets of K faulty modules - S_{ij} refers to the signature generated due to j^{th} fault in the i^{th} module M_i. The aim is to design a classifier which has to classify the signature sets into K classes, each class representing a module. The following example illustrates the formulation of circuit diagnosis as a pattern classification problem.

Let us consider the Example CUT ‘EC’ of Fig 15 with 5 POs (Primary Outputs). It has two partitions Module 1 and Module 2. The faulty signature set S_1 for Module 1 is computed through fault simulation by injecting each of the faults of this module. Similarly, the set S_2 is also computed for Module 2. The Signature Analyzer (SA) is a 5 bit maximum length CA or LFSR [8].

Let, faulty signature sets generated for two modules be $S_1 = \{0, 1, 8, 9, 16\}$ and $S_2 = \{2, 3, 7, 11, 31\}$. The MACA of Fig 1 represents the desired pattern classifier for the sets S_1 and S_2, where S_1 is covered by the zero attractor basin and S_2 by the non-zero attractor basins. The memory locations addressed by the pseudo-exhaustive field (*Theorem 2.1*) of attractors store the faulty module number.

At the testing phase, a faulty chip realizing ‘EC’ generates any one (say) signature 9 (01001). If we run the CA of Fig 1 with 9 as seed, it will reach to the attractor 00000 (Fig 1). By observing the PEF (00) of the attractor, the faulty module (Module 1) of ‘EC’, can be identified.

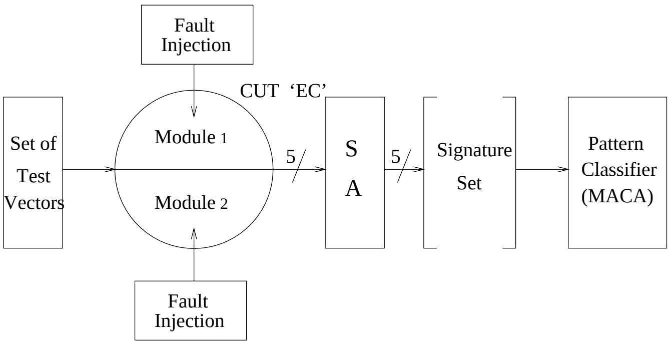


Figure 15. Diagnosis of an example CUT 'EC'

From the above example, it can be observed that the problem of diagnosis in electronic circuits maps to the design of MACA based Two Stage Classifier. The large fault dictionary gets replaced with the memory to store multiple Two Stage Classifiers of the multi-class classifier. The scenario of storing the fault dictionary is equivalent to storing the codebook for image compression as illustrated in Fig 12.

7.2. Experimental Results

In this subsection, we perform fault diagnosis of electronic circuits with MACA based Two Stage Classifier reported in *Section 4.* Fault diagnosis in both digital and analog circuits have been reported. While *Section 7.2.1* reports the diagnosis results for digital circuits, the same for analog circuits are covered in *Section 7.2.2.*

7.2.1. Digital Circuit Fault Diagnosis

The fault diagnosis experimentation for digital circuit has been carried out on a large number of benchmarks (ISCAS) for single stuck-at faults. A CUT (Circuit Under Test) is simulated with the fault simulator Hope [19]. The PO cones generated with Hope provide the basis of partitioning of the CUT. Hope extracts the PO cones to construct the fault list of a CUT. The fault list constructed by Hope reflects the spatial locality of wires in the CUT stuck at 1 or 0. The maximum length CA [8] is used to generate the signature of a fault in the CUT. The experimental setup, as shown in Fig 15, has been simulated in SUN with Solaris 5.6, 350 MHz clock.

Table 8 depicts the test environment and the circuit description. While Column I and II specify the benchmarks along with the number of internal lines, and Primary Input (PI), Output (PO) lines; Columns III and IV display the number of collapsed faults and the number of unique faulty signatures of the CUT. The Fault coverage figures noted in Column V are derived by applying a specific number of test patterns generated from a Pseudo Random Pattern Generator (PRPG) [8]. This testing experiment has been implemented only to demonstrate the elegance and efficiency of the proposed pattern classifier for fault diagnosis of electronic circuits.

The fault diagnosis results are presented in Table 9. The percentage of diagnosis achieved by the proposed scheme and with the fault dictionary method are noted in Column III(a) and III(b) respectively.

Table 8. Test results of diagnosis in dictionary method

CUT	No. of PI/PO Faults	No. of collapsed	No. of unique Faulty Signatures	Fault coverage
c880	60/26	942	896	98.62
c1908	33/25	1879	1801	98.99
c3540	50/22	3428	2838	96.00
c1355m	41/32	2278	1664	92.32
c1355	41/32	1574	912	98.48
c6288	32/32	7744	7648	99.50
c499m	41/32	902	776	96.01
c499	41/32	758	699	97.62
c2670	233/140	2747	2332	84.27
c7552	207/108	7550	7053	94.71
s967	16/23	1066	993	98.22
s991	65/17	910	776	95.27
s713	35/23	581	388	81.24
s3384	43/26	3380	2810	91.48
s4863	49/16	4764	4123	93.32
s3271	26/14	3270	2585	99.39
s5378	35/49	4603	2416	66.43
s6669	83/55	6684	6358	100
s641	35/24	467	359	85.22
c = combinational, s = sequential, m = mutant				

The percentage fault diagnosis is computed as

$$\% \text{ fault diagnosis} = \frac{\text{number of faults diagnosed}}{\text{number of detected faults}} \tag{9}$$

For the sake of illustration of diagnosis scheme, each of the circuits has been partitioned into a number of modules noted in Column II of Table 9. Diagnosis is done to the level of faulty module in the CUT. We have reported results for 4, 6, 8, 10 partitions for a CUT. Each partition represents a module. In Column IV, the memory requirement to implement the Two Stage Classifier is reported as a ratio of that required for diagnosis based on fault dictionary. For majority of the CUTs, the memory overhead is only a small percentage of the memory consumed by fault dictionary based scheme. This is due to the fact that MACA acts as an implicit memory. The diagnosis quality (Column III) and memory overhead figures (Column IV) establish the utility of the proposed pattern classifier for fault diagnosis of digital logic circuits.

7.2.2. Analog Circuit Fault Detection

A scheme is reported by Roy in [29] for synthesis of OTA (Operational Transconductance Amplifier) based analog circuits to realize different classes of filters. We have done experiment for diagnosis of faulty OTA and capacitor of such circuits.

The practical inability of testing analog circuits with desired precision demands the definition of a tolerance band within which the circuit functions correctly. This is particularly true when a parameter of the analog circuit (for example, transconductance of an OTA) cannot be tunned precisely as per the design requirement. As a result, test and diagnosis of an analog circuits has become a challenging problem. In general, test and diagnosis of an analog system demands analysis of a large volume of a test response data within the tolerance band [29]. We report the experimental setup for generation of faulty and fault free signatures for an analog circuit.

Table 9. Test results of MACA based Fault diagnosis

CUT (n, p)	No of	Fault Diagnosis (%)		Memory (%)
	Part^n	MACA	Dictionary	$\frac{\text{MACA}}{\text{Dictionary}} \times 100$
c880 (26, 896)	6	98.71	95.62	1.8243
c1908 (25, 1801)	6	98.83	96.03	0.9106
c3540 (22, 2838)	8	97.73	86.52	0.8184
c1355m (32, 1664)	6	98.10	78.94	0.9672
c1355 (32, 912)	6	97.75	58.84	1.7646
	8	96.71		2.4705
	10	93.55		3.1764
c6288 (32, 7648)	6	99.72	99.33	0.2104
c499m (32, 776)	10	99.31	91.72	3.7331
c499 (32, 699)	6	97.09	93.20	2.3024
c2670 (140, 2332)	10	100	99.19	1.2046
c7552 (108, 7053)	6	98.96	98.81	0.2238
	8	98.57		0.3134
	10	97.67		0.4029
s967 (23, 993)	4	98.28	94.08	0.9983
s991 (17, 776)	4	97.54	89.93	1.3189
s713 (23, 388)	4	93.62	81.85	2.5549
s3384 (26, 2810)	10	94.74	81.64	1.0471
s4863 (16, 4123)	10	92.43	89.63	0.7504
s3271 (14, 2585)	6	96.71	79.54	0.6648
	8	97.05		0.9308
	10	94.13		1.1968
s5378 (49, 2416)	4	95.03	76.21	0.3902
	8	98.06		0.9106
	10	92.35		1.1707
s6669 (55, 6358)	10	99.94	95.12	0.4426
s641 (24, 359)	6	95.86	88.86	4.5845
c = combinational, s = sequential, m = mutant				

Generation of Signatures

The set of signatures are derived out of the following data sets.

- $F_{good[i]}$ - data set of the good circuit for the i^{th} performance parameter, $i = 1, 2, \cdots, p$, where p denotes the maximum number of performance parameters considered.

- $F_{fault[ij]}$ - data set of the circuit by introducing fault at j^{th} device for i^{th} performance parameter. $j = 1, 2, \cdots, K$, where K is the number of components (OTA and capacitor) in the circuit.

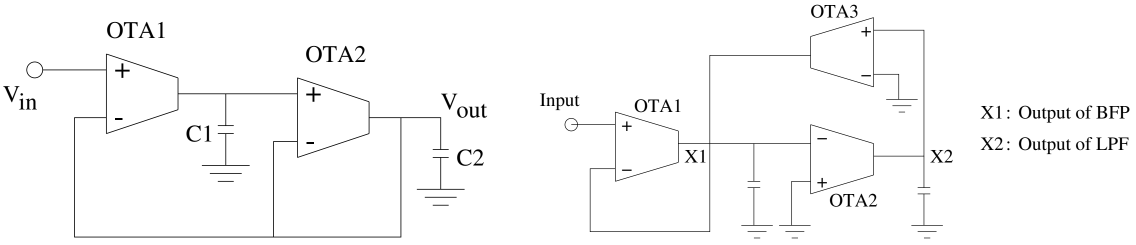


Figure 16. OTA realization of low pass and band pass filter

The discrete set of data - $F_{good[i]}$ and $F_{fault[ij]}$ has very large volume. It is compressed to a set of signatures $S = [S_0, S_1, \cdots, S_K]$, where S_0 represents the signature set for non-faulty circuit and S_x (for $x = 1, 2, \cdots, K$) denotes the signature set generated by introducing fault at x^{th} component, K being number of components. To generate the signature for the tolerance band (non-faulty region) and faulty region, a device parameter is divided into q number of equal steps. The corresponding signature is given as $S_x = [S_{(x)1}, S_{(x)2}, \cdots, S_{(x)q}]$, where $S_{(x)j}$ ($j = 1, 2, \cdots, q$) denotes the signature generated at the j^{th} step of the x^{th} component. The complete sequential steps for signature generation procedure are noted in [29].

The signature sets generated by the above procedure are partitioned into two classes - faulty and non-faulty. Faulty class is next divided into K number of sub-classes, each sub-class corresponds to signature set generated due to fault in a single component. The MACA based two class classifiers (Two Stage Classifier) have been synthesized at each level of diagnosis.

Experimental Results

In this subsection, we demonstrate the efficiency of the diagnosis procedure with two example circuits. Fig 16 show the OTA (Operational Transconductance Amplifier) realization of a low pass filter (LPF) and a band pass filter (BPF) respectively. The fault diagnosis results are shown in Table 10. While Column I specifies the type of filters, Column II shows the circuit components - OTAs and Capacitors. Column III specifies the number of sample values picked up from the output signal of the analog circuit. Digital signature is generated out of these digitized sample values. Faults detected under this experimental set up is shown in Column IV. The overall success rate of diagnosis of faulty component (OTA) is more than 99%. In both the cases (LPF and BPF), the memory requirement to implement Two Stage Classifier as a ratio of that required for diagnosis based on fault dictionary are 0.0372% and 0.0468% respectively. All the results reported in Table 10 confirm the high fault diagnosis efficiency and low memory overhead of the proposed Two Stage Classifier.

One of the major strength of the proposed classifier is its low cost high speed hardwired implementation for online real time application. So, we next deal with hardwired version of the MACA based pattern classifier.

Table 10. Fault Diagnosis Analysis of LPF and BPF of Fig 16

Filters	Compo- nents	No of Samples	Faults Detected	Faults not Detected	Success Rate
LPF	OTA1	8970	8962	8	99.91
	OTA2	5430	5424	6	99.88
	C1	5430	5421	9	99.83
	C2	5610	5603	7	99.87
BPF	OTA1	7201	7193	8	99.89
	OTA2	4321	4306	15	99.65
	OTA3	4441	4436	5	99.88
	C1	4297	4181	116	97.30
	C2	4321	4219	102	97.64

8. Hardware Architecture of Two Stage Classifier

In software implementation we employ Dependency Vector/String to identify the class of a pattern in O(n) time complexity. However, it is expensive to implement the associated logic of Dependency Vector/String in hardware compared to hardwired realization of Dependency Matrix of Linear CA (Table 1). So, to design the hardware of the proposed classifier, we synthesize a Dependency Matrix corresponding to each Dependency Vector/String representing an MACA. The following example illustrates this point.

Example 8.1. Dependency Matrix (T) corresponding to

1. $DV =< 1010111 >$
2. $DS = [< 10101 >< 0222 >]$

1. The Dependency Matrix for $DV =< 1010111 >$ is given by

$$T = \begin{bmatrix} |\overline{1} & \overline{1} & \overline{0}| & 0 & 0 & 0 & 0 \\ |0 & 1 & 1| & 0 & 0 & 0 & 0 \\ |\underline{0} & \underline{1} & \underline{1}| & \overline{1} & \overline{0}| & 0 & 0 \\ 0 & 0 & |0 & 1 & 1| & 0 & 0 \\ 0 & 0 & |\underline{0} & \underline{1} & \underline{1}| & \overline{1} & \overline{0}| \\ 0 & 0 & 0 & 0 & |0 & 0 & 1| \\ 0 & 0 & 0 & 0 & |\underline{0} & \underline{0} & \underline{0}| \end{bmatrix}$$

where the first block corresponds to $< 101 >$ followed by $< 101 >$ and $< 111 >$.

2. The DS consists of two DVs, where $DV_1 =< 10101 >$ and $DV_2 =< 0111 >$. Now, Dependency

Matrix for DV_1 is

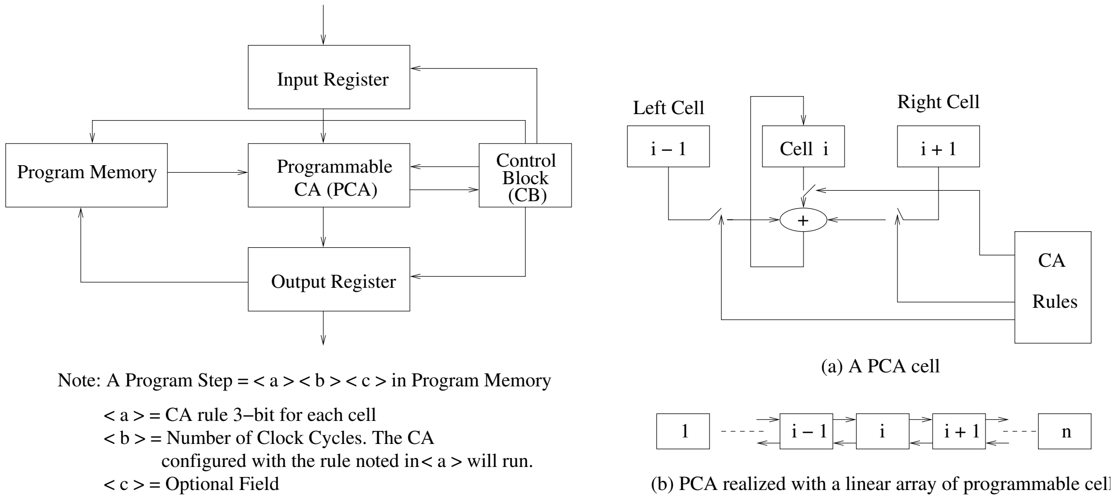
$$T_1 = \begin{bmatrix} \vert\overline{1} & \overline{1} & \overline{0}\vert & 0 & 0 \\ \vert 0 & 1 & 1\vert & 0 & 0 \\ \vert\underline{0} & \underline{1} & \underline{1}\vert & \overline{1} & \overline{0}\vert \\ 0 & 0 & \vert 0 & 1 & 1\vert \\ 0 & 0 & \vert\underline{0} & \underline{1} & \underline{1}\vert \end{bmatrix}$$

and Dependency Matrix for DV_2 is

$$T_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \vert\overline{1} & \overline{1} & \overline{0}\vert \\ 0 & \vert 0 & 0 & 1\vert \\ 0 & \vert\underline{0} & \underline{0} & \underline{0}\vert \end{bmatrix}$$

So, Dependency Matrix corresponds to Dependency String (DS) is given by

$$T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} = \begin{bmatrix} \vert\overline{1} & \overline{1} & \overline{0} & \overline{0} & \overline{0}\vert & 0 & 0 & 0 & 0 \\ \vert 0 & 1 & 1 & 0 & 0\vert & 0 & 0 & 0 & 0 \\ \vert 0 & 1 & 1 & 1 & 0\vert & 0 & 0 & 0 & 0 \\ \vert 0 & 0 & 0 & 1 & 1\vert & 0 & 0 & 0 & 0 \\ \vert\underline{0} & \underline{0} & \underline{0} & \underline{1} & \underline{1}\vert & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \vert\overline{0} & \overline{0} & \overline{0} & \overline{0}\vert \\ 0 & 0 & 0 & 0 & 0 & \vert 0 & 1 & 1 & 0\vert \\ 0 & 0 & 0 & 0 & 0 & \vert 0 & 0 & 0 & 1\vert \\ 0 & 0 & 0 & 0 & 0 & \vert\underline{0} & \underline{0} & \underline{0} & \underline{0}\vert \end{bmatrix}$$



Note: A Program Step = < a > < b > < c > in Program Memory

< a > = CA rule 3–bit for each cell
< b > = Number of Clock Cycles. The CA configured with the rule noted in< a > will run.
< c > = Optional Field

(a) A PCA cell

(b) PCA realized with a linear array of programmable cell

Figure 17. Hardware architecture of the proposed pattern classifier

The hardware architecture of the proposed MACA based pattern classifier is shown in Fig 17. A pattern classifier, as explained in *Section 4*, is built out of a multi-class classifier that is designed with a set of MACA based Two Stage Classifiers. Thus a pattern classifier consists of a set of MACAs. Each of these MACAs is a linear CA that employs CA rules noted in Table 1. To identify the class of an input element, it is necessary to traverse the MACA tree. So, basically we need to design a structure that can be configured as different MACAs at different time instances of the traversal. For hardwired design of a pattern classifier, we utilize the structure of a PCA (Programmable CA) proposed in [8]. The design of PCA and its application has been elaborated in [33]. A PCA supports a CA structure that can be programmed to realize different MACAs at different time instances. The structure of a PCA and its cell is noted in Fig 17. A PCA cell has 3 switches to configure any linear rule (Table 1) on a CA cell. Different MACA based two class classifier can be realized by programming such a PCA. The rule vectors of MACA realizing the pattern classifier are stored in Program Memory as shown in Fig 17. The input pattern is stored in the Input Register that initializes the PCA. The Control Block (CB) configures the PCA with the rule vector (field of a program step) and run for a number of cycles till an attractor state is reached. The CB controls the program flow - that is, it selects the next program step to be executed based on the output register value arrived at in the earlier program step. The process is continued until the program is fully executed that leads to identification of the class of the input pattern.

9. Conclusion

The paper presents the theory and application of an efficient pattern classifier, termed as Two Stage Classifier, which is built around a special class of sparse network referred to as Cellular Automata (CA). The proposed classifier reduces the complexity of the CA based classification algorithm from O(n^3) (as reported in [12]) to O(n). Extensive experimental results have been reported in respect of its application in the fields of data mining, image compression, and fault diagnosis. The excellent classification accuracy and low memory overhead figures prove the superiority of the CA based classifier over that of the existing classification algorithms. The simple structure of CA makes it ideally suitable for VLSI implementation.

Acknowledgement

This research work is supported by AICTE sponsored project F. No. 8020/RID/TAPTEC-46/2001-2002. We acknowledge AICTE, India. We would also like to thank the anonymous referees for providing helpful comments and valuable criticisms which have greatly improved the presentation of the paper.

References

[1] Abramovici, M., Breuer, M., Friedman, A.: Digital Systems Testing and Testable Design, *IEEE Press*, 1990.

[2] Agrawal, R., Imielinski, T., Swami, A.: Database Mining: A Performance Perspective, *IEEE Transaction on Knowledge and Data Engineering*, **5**(6), December 1993.

[3] Agrawal, V. D., Mercer, M. R.: Testability Measures - What Do They Tell Us?, *In Proc. International Test Conference*, 1982, 391–396.

[4] Bardell, P., McAnney, W., Savir, J.: Built In Test for VLSI: Pseudorandom Techniques, *Wiley Interscience*, 1987.

[5] Boppana, V., Fuchs, W. K.: Fault Dictionary Compaction by Output Sequence Removal, *In ICCAD*, November 1994, 576–579.

[6] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and Regression Trees, *Wadsworth, Belmont*, 1984.

[7] Chattopadhyay, S., Adhikari, S., Sengupta, S., Pal, M.: Highly Regular, Modular, and Cascadable Design of Cellular Automata-Based Pattern Classifier, *IEEE Transaction on VLSI Systems*, **8**(6), December 2000, 724–735.

[8] Chaudhuri, P. P., Chowdhury, D. R., Nandi, S., Chatterjee, S.: Additive Cellular Automata, Theory and Applications, VOL. 1, *IEEE Computer Society Press, Los Alamitos, California*, **ISBN-0-8186-7717-1**, 1997.

[9] Cheeseman, P., Stutz, J.: Bayesian Classification (AutoClass): Theory and Results, *In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press*, 1996, 153–180.

[10] Gallant, S. I.: Neural Networks Learning and Expert Systems, *Cambridge, Mass; MIT Press*, 1993.

[11] Ganguly, N.: Cellular Automata Evolution : Theory and Applications in Pattern Recognition and Classification, *Ph.D Thesis, CST Dept, BECDU, India*, 2003.

[12] Ganguly, N., Maji, P., Dhar, S., Sikdar, B. K., Chaudhuri, P. P.: Evolving Cellular Automata as Pattern Classifier, *Proceedings of Fifth International Conference on Cellular Automata for Research and Industry, ACRI 2002, Switzerland*, October 2002, 56–68.

[13] Gantamatcher, F. R.: The Theory of Matrices, *Chelsa Publishing Co., New York,* **I & II**, 1959.

[14] Goldberg, D. E.: Genetic Algorithms in Search, Optimizations and Machine Learning, *Morgan Kaufmann*, 1989.

[15] Gresho, A., Gray, R. M.: Vector Quantization and Signal Compression, *Kluwer Academy Publishers*, 1992.

[16] Han, J., Kamber, M.: Data Mining, Concepts and Techniques, *Morgan Kaufmann Publishers*, **ISBN : 1-55860-489-8**, 2001.

[17] Hertz, J., Krogh, A., Palmer, R. G.: Introduction to the Theory of Neural Computation, *Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley*, 1991.

[18] Holland, J. H.: Adaptation in natural and artificial systems, *The University of Michigan Press, Ann Arbor, MI*, 1975.

[19] Lee, H. K.: Hope: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits, *In Proceedings of* 29^{th} *Design Automation Conference*, 1992, 336–340.

[20] Lippmann, R.: An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, **4(22)**, April 1987.

[21] Matheus, C. J., Chan, P. K., Piatetsky-Shapiro, G.: Systems for Knowledge Discovery in Databases, *IEEE Transaction on Knowledge and Data Engineering*, **5**, 1993, 903–913.

[22] Mehta, M., Agrawal, R., Rissanen, J.: SLIQ: A Fast Scalable Classifier for Data Mining, *Proceedings of International Conference on Extending Database Technology, Avignon, France*, 1996.

[23] Neumann, J. V.: The Theory of Self-Reproducing Automata, *A. W. Burks, Ed. University of Illinois Press, Urbana and London*, 1966.

[24] Pal, K.: Theory and Application of Multiple Attractor Cellular Automata for Fault Diagnosis, *Proceedings of Asian Test Symposium*, December 1998.

[25] Piatetsky-Shapiro, G., Fayyad, U., Smith, P.: From Data Mining to Knowledge Discovery: An Overview, *In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press*, 1996, 1–35.

[26] Piatetsky-Shapiro, G., Frawley, W. J.: Knowledge Discovery in Databases, *AAAI/MIT Press*, 1991.

[27] Pomeranz, I., Reddy, S. M.: On The Generation of Small Dictionaries for Fault Location, *In ICCAD*, November 1992, 272–279.

[28] Quinlan, J. R.: C4.5: Programs for Machine Learning, *Morgan Kaufmann, CA*, 1993.

[29] Roy, B. N., Chaudhuri, P. P., Nandi, P. K.: Efficient Synthesis of OTA Network for Linear Analog Functions, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, **21**(5), May 2002, 517–533.

[30] Ryan, P. G., Fuchs, W. K., Pomeranz, I.: Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits, *In ICCAD*, November 1993, 508–511.

[31] Santa-Cruz, D., Ebrahimi, T., Askelf, J., Larsson, M., Christopoulos, C.: JPEG 2000 still image coding versus other standards, *Proceedings of the SPIE's 46th annual meeting, Applications of Digital Image Processing XXIV, San Diego, California*, **4472**, July 2001, 267–275.

[32] Shafer, J., Agrawal, R., Mehta, M.: SPRINT: A Scalable Parallel Classifier for Data Mining, *In 22nd VLDB Conference*, 1996.

[33] Shaw, C., Chatterjee, D., Maji, P., Sen, S., Chaudhuri, P. P.: A Pipeline Architecture For Encompression (Encryption + Compression) Technology, *Proceedings of* 16^{th} *International Conference on VLSI Design, India*, January 2003.

[34] Sikdar, B. K., Ganguly, N., A.Karmakar, S.Chowdhury, Chaudhuri, P. P.: Multiple Attractor Cellular Automata for Hierarchical Diagnosis of VLSI Circuits, *Proceedings of* 10^{th} *Asian Test Symposium, Japan*, 2001.

[35] Sikdar, B. K., Ganguly, N., Majumder, P., Chaudhuri, P. P.: Design of Multiple Attractor $GF(2^p)$ Cellular Automata for Diagnosis of VLSI Circuits, *Proceedings of* 14^{th} *International Conference on VLSI Design, India*, January 2001, 454–459.

[36] Wolfram, S.: Theory and Application of Cellular Automata, *World Scientific*, 1986.