Design of SMACA: synthesis and its analysis through rule vector graph for web based application

Anirban Kundu*, Ruma Dutta and Debajyoti Mukhopadhyay

Web Intelligence and Distributed Computing Research Lab,
Techno India Group,
West Bengal University of Technology,
EM 4/1, Sector V, Salt Lake, Calcutta 700091, India
E-mail: anik76in@gmail.com
E-mail: rumadutta2006@gmail.com
E-mail: debajyoti.mukhopadhyay@gmail.com
*Corresponding author

Abstract: Web search engine uses indexing for management of web-pages in a mannered way. Web-pages are well distributed within the database of server. Both forward and inverted indexing is employed to tackle web-pages as a part of its functional design. This indexing mechanism helps in retrieving data from the database based on user query. In this paper, an efficient solution to handle the indexing problem is proposed with the introduction of non-linear single cycle multiple attractor cellular automata (SMACA). This paper also reports an analysis on SMACA using rule vector graph (RVG). This work simultaneously shows generation of SMACA by using specific rule sequence. Searching mechanism is done with O(n) complexity. SMACA provides an implicit memory to store the patterns. Search operation to identify the class of a pattern out of several classes boils down to running a cellular automata (CA) for one time step. This demands storage of the CA rule vector (RV) and the seed values. SMACA is based on sound theoretical foundation of CA technology.

Keywords: search engine; single cycle multiple attractor cellular automata; SMACA; World Wide Web; WWW; indexing storage; rule vector graph; RVG.

Reference to this paper should be made as follows: Kundu, A., Dutta, R. and Mukhopadhyay, D. (2008) 'Design of SMACA: synthesis and its analysis through rule vector graph for web based application', *Int. J. Intelligent Information and Database Systems*, Vol. 2, No. 4, pp.397–421.

Biographical notes: Anirban Kundu is doing his research work at Web Intelligence and Distributed Computing Research Lab (WIDiCoReL) under Techno India Group (TIG). He is also working as a Lecturer in Information Technology (IT) Department of Netaji Subhash Engineering College, Garia, Kolkata, West Bengal, India.

Ruma Dutta is doing her research work at Web Intelligence and Distributed Computing Research Lab (WIDiCoReL) under Techno India Group (TIG). She is also working as a Senior Lecturer in Computer Science and Engineering (CSE) Department of Netaji Subhash Engineering College, Garia, Kolkata, West Bengal, India.

Debajyoti Mukhopadhyay is the Founder Director of the Web Intelligence and Distributed Computing Research Lab (WIDiCoReL) and is a Professor of Computer Science and Engineering and Information Technology at Techno India Group of Engineering Colleges (West Bengal University of Technology) for the BTech, MTech and PhD programs. He is also the Head of the MBA (systems) program at TIG Business Schools. He has been working as a Full Professor of Computer Science and Engineering at the West Bengal University of Technology affiliated engineering colleges since 2001.

1 Introduction

The researchers in artificial life inherit the tradition and epistemology of simulation. They model the world as a collection of dynamical systems, complex and non-linear in nature. They hold the view that the universe is a computer implementing transformations of information. If the universe can be viewed as a computation, it should be possible to build computing models of physical systems of the universe. Many researchers in the field of artificial life have been enamoured of a mathematical formalism of computing model known as cellular automata (CA).

Most people today can hardly imagine life without the internet (Flake et al., 2000; Glover et al., 2002). Kundu et al. (2007) reported a shorter version of this work. It provides access to information, news, e-mail, shopping and entertainment. World Wide Web (WWW) has brought huge information at the door-step of every user. The World Wide Web Worm (WWWW) was one of the first web search engines which was basically a storage of huge volume of information. To handle these information, proper indexing has been done in several ways (Brin and Page, 1998; Arasu et al., 2001).

This work reports an efficient scheme for designing an n-cell single cycle multiple attractor cellular automata (SMACA) (Sipper, 1996; Maji et al., 2003) for handling the forward indexing and inverted indexing in a fast and inexpensive way. It is built around non-linear scheme. Generated SMACAs have been used for information storage which requires special attention considering the huge volume of data in web to be dealt with by the search engines (Chakrabarti et al., 1999; Mukhopadhyay and Singh, 2004). The major contributions of this paper can be summarised as follows:

1 synthesis of SMACA

2 analysis of SMACA through rule vector graph (RVG)

3 usage of SMACA in forward indexing

4 usage of SMACA for replacing inverted indexing

5 searching mechanism using SMACA

6 experimental results.

2 CA preliminaries

An *n* cell CA consists of *n* cells [Figure 1(a)] with local interactions (Neumann, 1966). It evolves in discrete time and space. The next state function of three neighbourhood CA cell [Figure 1(b)] can be represented as a rule as defined in Table 1 (Wolfram, 1986). First row of Table 1 represents $2^3 = 8$ possible present states of three neighbours of *i*th cell – (i–1), i, (i+1) cells. Each of the eight entries (3-bit binary string) represents a minterm of a three variable boolean function for a three neighbourhood CA cell. In subsequent discussions, each of the eight entries in Table 1 is referred to as a rule min term (RMT). The decimal equivalent of eight minterms are 0, 1, 2, 3, 4, 5, 6, 7 noted within ( ) below the three bit string. Each of the next five rows of Table 1 shows the next state (0 or 1) of *i*th cell. Hence, there can be $2^8 = 256$ possible bit strings. The decimal counterpart of such an 8-bit combination is referred to as a CA rule (Wolfram, 1986; Chaudhuri et al., 1997). The rule of a CA cell can be derived from Table 1 of the *i*th cell.

Figure 1 Local interaction between CA cells (a) an n cell CA with null boundary and (b) the *i*th cell configured with rule R_i

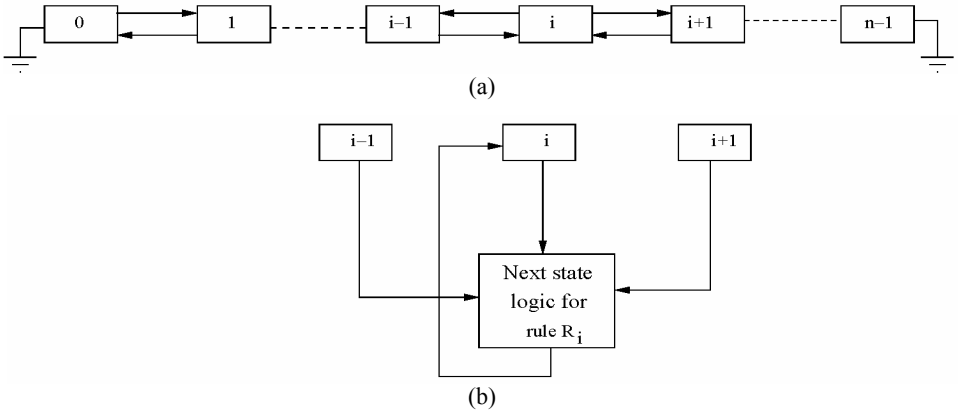


Table 1 Truth table of sample rules of a CA cell showing the next state logic for the minterms of a three variable boolean function

Present states of (i−1), i and (i+1) cells	*111 (7)*	*110 (6)*	*101 (5)*	*100 (4)*	*011 (3)*	*010 (2)*	*001 (1)*	*000 (0)*	*Rule*
Next state of *i*th cell	0	1	0	1	1	0	1	1	90
	1	0	0	1	0	1	1	0	150
	0	1	1	1	1	0	0	0	120
	0	0	0	0	1	1	0	0	12
	1	1	0	1	0	0	1	0	210

Note: The eight minterms having decimal values 0, 1, 2, 3, 4, 5, 6, 7 are referred to as rule minterms (RMTs).

Definition 1 Group CA – each state in the state transition behaviour of a group CA has only one predecessor and consequently each state is reachable from only one state. A group CA traverses all the states in a cycle. A group CA is a reversible CA in the sense that the CA will always return to its initial state.

Definition 2 Non-group CA – a non-group CA has states that have r number of predecessors, where $r = 0, 1, 2, 3, \ldots$

Definition 3 Reachable state – a state having one or more predecessors is a reachable state.

Definition 4 Non-reachable state – a state having no predecessor (that is, r = 0) is termed as non-reachable.

Definition 5 Transient state – a non-cyclic state of a non-group CA is referred to as a transient state.

Definition 6 Attractor cycle – the set of states in a cycle is referred to as an attractor cycle.

Definition 7 Self-loop attractor (SLA) – a single cycle attractor state with self-loop is referred to as SLA.

Definition 8 Rule vector (RV) – the sequence of rules $< R_0 R_1 \cdots R_i \cdots R_{n-1} >$, where ith cell is configured with rule R_i, is known as RV.

3 Existing mechanism of a typical search engine

Figure 2 Schematic diagram of a typical search engine

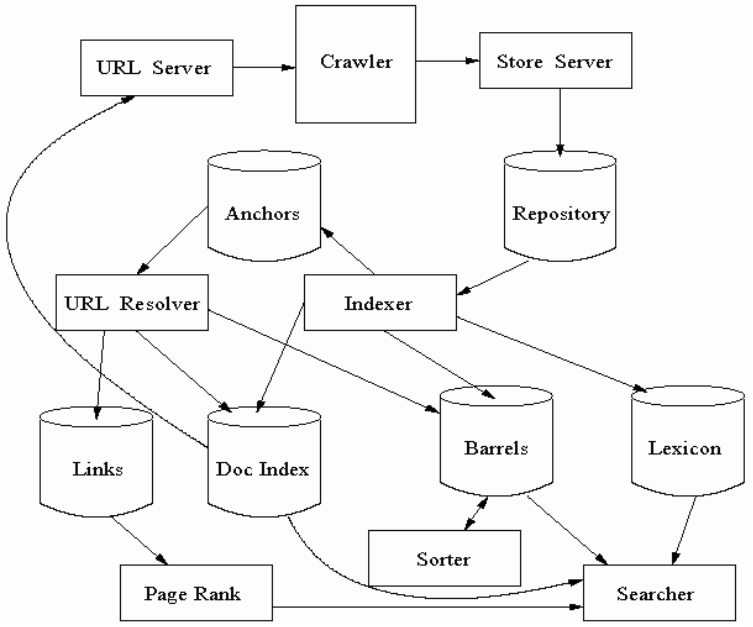


Figure 2 shows a search engine schematically. Every engine depends on crawler module. Crawlers browse the web on the search engine's behalf. Using initial set of URLs, the related pages are retrieved from the web. The crawlers extract URLs appearing in the retrieved pages and pass on the information to the crawler control module. This module decides which links are to be visited next. The crawlers pass the retrieved pages into a page repository. The indexer module extracts all the words from each retrieved page and records the URL where each word occurred in form of forward index. Forward index is sorted on document identification number (docID) which is assigned whenever a new URL is parsed out of a web-page. This forward index is converted into inverted indexed file which is again sorted on the basis of word identification number (wordID) (Brin and Page, 1998; Arasu et al., 2001).

4 Generation of SMACA

Synthesis of SMACA demands formation of a RV with group and non-group rules in specific sequence. The method to identify such a sequence is described in the following discussions. A scheme is outlined here to identify the sequence of rules in the RV that makes the CA a SMACA. The RV of an n-cell CA is denoted as $< R_0, R_1 \cdots R_i, R_{i+1}, \cdots, R_{n-1} >$, where ith cell is configured with R_i. A non-linear (Das et al., 2003, 2004) SMACA consists of 2^n number of states where n is the size of SMACA. The structure of a non-linear SMACA has attractors (self-loop or single length cycle), non-reachable states and transient states. The attractors form unique classes (basins). All other states reach the attractor basins after certain time steps. To classify a set of k classes, (k – 1) number of attractors are used, each identifying a single class. Consider, k = 4 for a particular situation, i.e., four attractors are required. To manage this situation, '00', '01', '10' and '11' may be considered as attractors for classification of distinct states into four categories. Instead of using four attractors, three attractors may be used. So, we may consider '00', '01', '10' as attractors and the 4th attractor need not be specified. If we put concerned states over these three attractors, remaining states can be considered under the unspecified (4th) attractor. Chaudhuri et al. (1997) and Maji et al. (2003) reported illustrative idea in this matter. Figure 3 shows an arbitrary example of non-linear SMACA with its irregular structure. States 1 and 9 are attractors. States 3, 5, 7, 11, 13 and 15 are transient states. All other states are non-reachable states.

It is found through exhaustive experimentation that there are 15 such classes for all the rules which can be used to form SMACA in a specific sequence. These classes are denoted as {I, II, III, IV, V, VI, VII, VIII, IX, X, XI, XII, XIII, XIV and XV} in Table 2. Table 3 dictates the rule of $(i + 1)$th cell from the class of ith cell. The table is formed in such a way that SMACA is formed if and only if the relationship between R_i and R_{i+1} is maintained. Since the design is concerned with null boundary CA, there are $2^{2^2} = 16$ effective rules for the left most (R_0) as well as the right most (R_{n-1}) cells. The RMTs 4, 5, 6 and 7 can be treated as don't care for R_0 as the present state of left neighbour of cell 1 is always 0. So, there are only four effective RMTs (0, 1, 2 and 3) for R_0. Similarly, the RMTs 1, 3, 5 and 7 are don't care RMTs for R_{n-1}. The effective RMTs for R_{n-1} are 0, 2, 4 and 6. R_0 and R_{n-1} are listed in Table 4 and Table 5 respectively.

Theorem 4.1 *A specific sequence of group and/or non-group rules forms a SMACA.*

Let us consider a CA with RV $R =< R_0, R_1 \cdots R_i, R_{i+1}, \cdots, R_{n-1} >$, where all the R_is are either group or non-group rules. The local CA rules R_is can be so configured that if a CA is loaded with any seed, produces two types of states $\{\cdots d_i d_{i+1} \cdots\}$ and $\{\cdots d_i' d_{i+1}' \cdots\}$. The d_i (= 0/1) is the state of ith cell and d_i' is its complement. Therefore, there are 2^n number of current states for which the next states take the form $S = \left\{\cdots d_i d_{i+1} \cdots, \cdots d_i' d_{i+1}' \cdots\right\}$. The maximum possible cardinality of S is $2*2^{n-2} = 2^{n-1}$. Since the number of next states is lesser than that of current states, there exists at least a state in S with more than one predecessor. Therefore, the CA is non-group which means CA may be a SMACA or may be a general non-group CA. Hence, any sequence of group and/or non-group rules can not form a SMACA.

The proof of the following Lemmas follows from the above discussion.

Lemma 4.2 If S_n is the set of next states of a CA with S_c as the set of current states, then the CA is group iff $||S_c|| = ||S_n||$.

Lemma 4.3 If S_n is the set of next states of a CA with S_c as the set of current states, then the CA is non-group iff $||S_c|| \neq ||S_n||$.

Therefore, the synthesis of a SMACA demands formation of a RV with group and/or non-group rules in specific sequence. The method to identify such a sequence is described in the following sub-section.

Figure 3 Structure of a SMACA with RV <4 102 53 85>

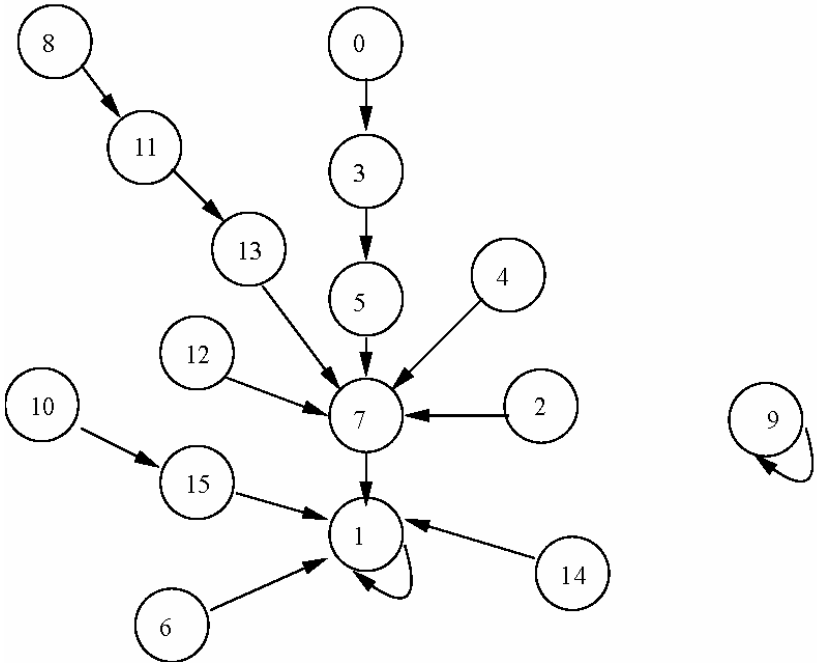


Table 2 SMACA class table

Class	*Rules*
I	0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240
II	1, 17, 33, 49, 65, 81, 97, 113, 129, 145, 161, 177, 193, 209, 225, 241
III	2, 18, 34, 50, 66, 82, 98, 114, 130, 146, 162, 178, 194, 210, 226, 242
IV	4, 20, 36, 52, 68, 84, 100, 116, 132, 148, 164, 180, 196, 212, 228, 244
V	5, 21, 37, 53, 69, 85, 101, 117, 133, 149, 165, 181, 197, 213, 229, 245
VI	6, 22, 38, 54, 70, 86, 102, 118, 134, 150, 166, 182, 198, 214, 230, 246
VII	7, 23, 39, 55, 71, 87, 103, 119, 135, 151, 167, 183, 199, 215, 231, 247
VIII	8, 24, 40, 56, 72, 88, 104, 120, 136, 152, 168, 184, 200, 216, 232, 248
IX	9, 25, 41, 57, 73, 89, 105, 121, 137, 153, 169, 185, 201, 217, 233, 249
X	10, 26, 42, 58, 74, 90, 106, 122, 138, 154, 170, 186, 202, 218, 234, 250
XI	11, 27, 43, 59, 75, 91, 107, 123, 139, 155, 171, 187, 203, 219, 235, 251
XII	12, 28, 44, 60, 76, 92, 108, 124, 140, 156, 172, 188, 204, 220, 236, 252
XIII	13, 29, 45, 61, 77, 93, 109, 125, 141, 157, 173, 189, 205, 221, 237, 253
XIV	14, 30, 46, 62, 78, 94, 110, 126, 142, 158, 174, 190, 206, 222, 238, 254
XV	15, 31, 47, 63, 79, 95, 111, 127, 143, 159, 175, 191, 207, 223, 239, 255

4.1 Synthesis of SMACA

The synthesis algorithm generates the RV $R =< R_0, R_1 \cdots, R_{n-1} >$ for an n-cell SMACA, where R_i is the rule with which the ith CA cell is to be configured. The characterisation of SMACA points to the fact that the design of SMACA for any arbitrary n boils down to:

1 Form the classes of rules – that is, formation of Table 2 to Table 5.

2 Find the class of (i+1)th cell rule depending on the rule of ith cell and its class.

Task 1 Construction of Table 2 to Table 5 involves one time cost.

Task 2 The class of (i + 1)th cell rule is determined from the rule R_i and its class. Based on the rule class table (Table 2 to Table 5), we sequentially assign a rule R_i+1 to the (i + 1)th CA cell (i = 1, 2,…, (n–1)) to form the RV $R =< R_0, R_1 \cdots R_i, \cdots, R_{n-1} >$. The R_0 is selected randomly from Table 4 and R_{n-1} from Table 5. Based on Task 2, Algorithm 1 is further designed.

Table 3 Relationship of R_i and R_{i+1}

Class of R_i	R_{i+1}
I	0–2, 4–18, 20–34, 36–50, 52–66, 68–82, 84–98, 100–114, 116–130,132–146, 148–162, 164–178, 180–194, 196–210, 212–226, 228–242, 244–255
II	20, 22, 25, 28, 29, 30, 38, 40–41, 44–46, 52, 54, 56–57, 60–62, 69, 71, 75, 77, 79, 84–87, 89, 91–95, 101–111, 116–127, 135, 138–139, 142–143, 148–151, 153–159, 166–175, 180–191, 197, 199, 202–203, 205–207, 212–215, 217–223, 229–239, 244–255
III	0–2, 4–6, 8–10, 12–14, 16–18, 20–22, 24–26, 28–30, 32–34, 36–38, 40–42, 44–46, 52, 54, 56–57, 60–62, 64–66, 68–70, 72–74, 76–77, 80–82, 84–86, 88–89, 92–93, 96–98, 100–102, 104–106, 108–109, 116, 118, 120–121, 124–125, 128–130, 132–134, 136–138, 140, 142, 144–146, 148–150, 152–154, 156, 158, 160–162, 164, 166, 168–170, 172, 174, 180, 182, 184–185, 188, 190, 192–194, 196–197, 200, 202, 208–210, 212–213, 224–226, 232, 234
IV	0–2, 4–18, 20–34, 36–50, 52–66, 68–82, 84–98, 100–114, 116–130, 132–146, 148–162, 164–178, 180–194, 196–210, 212–226, 228–242, 244–255
V	0–2, 4–6, 8–10, 12–14, 16–18, 20–22, 24–26, 28–29, 32–34, 36–38, 40–42, 44, 46, 64–66, 68–74, 76–82, 84–96, 98, 100–111, 116–119, 122–130, 132–134, 136–145, 148–162, 164–175, 181, 183–194, 196–209, 212–224, 226, 228–239, 244–255
VI	0–14, 16–26, 28–30, 32–38, 40–46, 48–50, 52–54, 56–58, 60–62, 64–77, 80–89, 92–93, 96–102, 104–109, 112–113, 116–117, 120–121, 124–125, 128–140, 142, 144–154, 156, 158, 160–164, 166, 168–172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192–203, 208–215, 224–227, 232–235
VII	0–2, 4–6, 8–10, 12–14, 16–18, 20–22, 24–26, 28–30, 32–34, 36–38, 40–42, 44–46, 64–77, 80–82, 84–86, 88–89, 92–93, 96–107, 128–140, 142, 144–155, 160–162, 164, 166, 168–170, 174, 192–203, 208–215, 224–227, 232–235
VIII	0–2, 4–18, 20–34, 36–50, 52–66, 68–82, 84–98, 100–114, 116–130, 132–146, 148–162, 164–178, 180–194, 196–210, 212–226, 228–242, 244–255
IX	20–23, 28–31, 40–47, 52–63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83–87, 89, 91–95, 97, 99, 101–111, 113, 115–127, 130–131, 134–135, 138–139, 142–143, 146–151, 153–159, 162–163, 166–175, 178–191, 193–195, 197–199, 201–203, 205–207, 209–215, 217–223, 225–227, 229–239, 241–255
X	0–2, 4–6, 8–10, 12–14, 16–18, 20–21, 24–26, 28–29, 32–34, 36–38, 40, 42, 44, 46, 64–66, 68–74, 76–82, 84–98, 100–106, 108–111, 116–119, 121–134, 136–146, 148–150, 152–162, 164–175, 181–194, 196–209, 212–224, 226, 228–239, 244–255
XI	65, 67, 69, 71, 73, 75, 77, 79, 84–87, 89, 91–95, 97–99, 101–103, 105–107, 109–111, 116–127, 130–131, 134–135, 138–139, 142–143, 145–147, 149–151, 153–155, 157–159, 166–175, 180–191, 193–195, 197–199, 201–203, 205–207, 209–215, 217–223, 225–227, 229–239, 244–255
XII	0–2, 4–17, 20–21, 24–32, 34, 36–40, 42, 44–47, 64–66, 68–82, 84–96, 98, 100–104, 106, 108–112, 114, 116–120, 122, 124–130, 132–145, 148–149, 152–162, 164–177, 180–181, 184–194, 196–210, 212–226, 228–242, 244–255
XIII	0–47, 64–255
XIV	0–47, 64–255
XV	0–47, 64–255

Table 4 First rule table (R_0)

Rules for R_0	*Class of* R_0
0	I
1	II
2	III
4	IV
7	VII
8	VIII
11	XI
13	XIII
14	XIV
15	XV

Table 5 Last rule table (R_{n-1})

Class for R_{n-2}	*Rules for* R_{n-1}
I	0, 4, 16, 21, 64, 69, 84, 85
II	69, 84, 85
III	0, 4, 64
IV	0, 4, 16, 21, 64, 69, 84, 85
V	0, 4, 64, 69, 84, 85
VI	0, 1, 4, 16, 64
VII	0, 4, 64
VIII	0, 4, 16, 21, 64, 69, 84, 85
IX	21, 69, 81, 84, 85
X	0, 4, 64, 69, 84, 85
XI	69, 84, 85
XII	0, 4, 64, 69, 84, 85
XIII	0, 1, 4, 64, 69, 81, 84, 85
XIV	0, 1, 4, 64, 69, 81, 84, 85
XV	0, 1, 4, 64, 69, 81, 84, 85

For the formation of SMACA, the synthesis scheme is achieved through Algorithm 1.

Algorithm 1: SMACA synthesis

Input n (CA size), Tables (2, 3, 4 and 5)

Output a SMACA – that is, RV $R =< R_0, R_1, \cdots, R_{n-1} >$

Step 1 Pick up the first rule R_0 randomly from Table 4 and check the class of R_0
C := Class of R_0 (C $\in$ {I, II, III, IV, VII, VIII, XI, XIII, XIV, XV} of Table 4)

Step 2 For i := 1 to n–2; repeat Step 3 and Step 4

Step 3 From Table 3 pick up a rule as R_{i+1} arbitrarily for Class C

Step 4 Find Class C for the current cell rule using Table 2

Step 5 From Table 5 pick up a rule as R_{n-1}

Step 6 Form the RV $R =< R_0, R_1, \cdots, R_{n-1} >$

Step 7 Stop

The complexity of Algorithm 1 is O(n).

Example 1 Synthesis of 4-cell SMACA:

Consider, rule 2 is selected as R_0. Therefore, the class (obtained from Table 4) of current cell rule is III. From class III of Table 3, rule 44 is selected randomly as R1. Find class of rule 44 from Table 2. Since, rule 44 is of class XII; so, from Table 3, the next state value can be easily found by selecting a random value as r^{th}_{i+1} rule. Say, rule 4 is selected as R_2. From Table 2, rule 4 is of class IV. Rule R_{n-1} is selected based on the class of R_{n-2} from Table 5. So, rule 21 is selected randomly for R_3 from Table 5 as R_{n-1}. Therefore, the SMACA is R = <2, 44, 4, 21>.

4.2 Analysis of SMACA through RVG

Definition 9 Non-group rule – a rule is a non-group rule if its presence in a RV makes the CA non-group.

Definition 10 Group rule – a rule is a group rule if it is not a non-group rule.

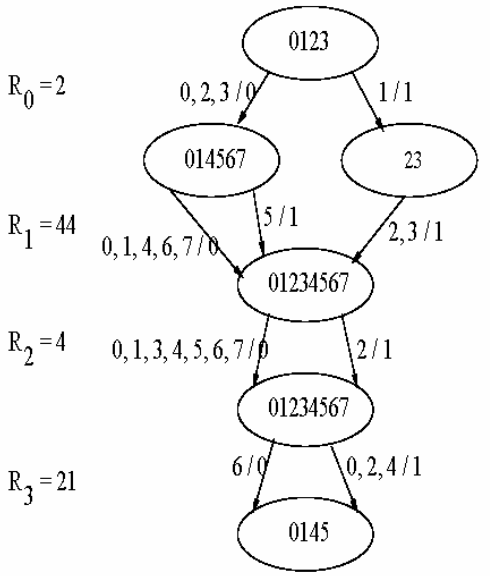
Definition 11 Balanced rule – a rule is balanced if it contains equal number of 1s and 0s in its 8-bit binary representation.

Definition 12 Unbalanced rule – a rule is unbalanced if it is not balanced.

Definition 13 RVG – a graph, derived from the RV of a CA is referred to as RVG.

Figures 4 and 5 are the representation of SMACA in RVG form. RVG consists of several nodes with one or more links. Node values are basically the RMT values of specific rules. A RVG always starts with a single node containing '0', '1', '2' and '3' as node value. Since we have used three-neighbourhood null boundary CA, the R_0 and R_{n-1} have four don't care positions in its concerned RMTs. So, in case of R_0 the effective RMT positions are 0, 1, 2 and 3. Similarly, for R_{n-1}, the effective RMT positions are 0, 2, 4 and 6 as discussed earlier in Section 4. In three-neighbourhood null boundary CA, RMT values range from (0–7). So, maximum eight values are there within the RVG. Any combination of (0–7) can be a node value depending on the applied rule on that particular level of RVG. Each node can have maximum of two edges (0-edge and 1-edge). The reason behind this is each cell's next state value may be either '0' or '1'. So, depending on the probability to appear '0' or '1' as the next state value, RMTs of each node is divided into two categories. So, at any level of graph, maximum of two nodes can be formed from each node of previous level.

Figure 4 RVG and related RMT of a 4-cell CA with RV <2, 44, 4, 21>



	Rule min term (RMT)							
	111	110	101	100	011	010	001	000
2	x	x	x	x	0	0	1	0
44	0	0	1	0	1	1	0	0
4	0	0	0	0	0	1	0	0
21	x	0	x	1	x	1	x	1

Following statements have been achieved through exhaustive experimentation and corresponding analysis of RVG:

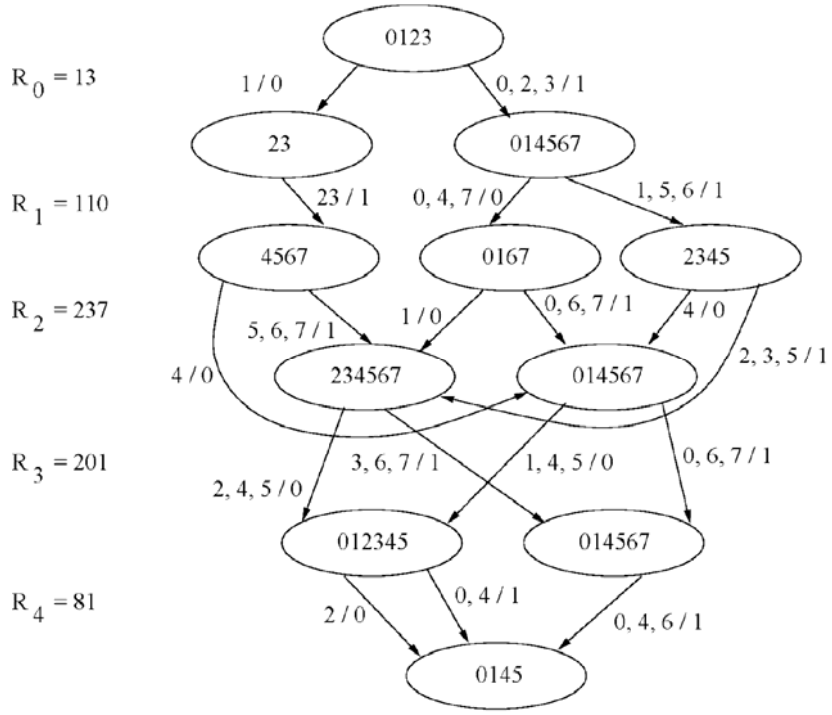
1 Root node has maximum four RMT values; since R_0 of a three-neighbourhood null boundary CA can have maximum of four RMTs 0, 1, 2 and 3.

2 Maximum two nodes can be generated by 0-edge and 1-edge of a node as output nodes in next level of RVG containing Type-1/Type-2 of Class-1/Class-2/Class-3, or a subset of any of these classes of RMTs, or a mixture of a full RMT class value and a subset of another class of RMT. In Table 6, all the class values of RMT have been shown.

3 At any level of RVG of group/non-group CA, minimum number of nodes = 1; and, maximum number of nodes = 4.

4 Each input RMT value (0–7) generates two probable output RMT values through left shift operation. The detailed mapping is shown in Table 7.

Figure 5 RVG and related RMT of a 5-cell CA with RV <13, 110, 237, 201, 81>



	Rule min term (RMT)							
	111	110	101	100	011	010	001	000
13	x	x	x	x	1	1	0	1
110	0	1	1	0	1	1	1	0
237	1	1	1	0	1	1	0	1
201	1	1	0	0	1	0	0	1
81	x	1	x	1	x	0	x	1

Table 6 Generic node values of RVG for even distribution of RMTs on edges

Class name	*Type-1*	*Type-2*
Class-1	0123	4567
Class-2	0145	2367
Class-3	0167	2345

Table 7 Probable next level RMTs

Input RMTs	*Output RMTs*
0, 4	0, 1
1, 5	2, 3
2, 6	4, 5
3, 7	6, 7

Table 8 Generic node values of RVG for un-even distribution of RMTs on edges

Class Name	*Type 1*	*Type 2*
Class-1	01234567	
Class-2	01	234567
Class-3	23	014567
Class-4	45	012367
Class-5	67	012345

Theorem 4.4 *At any level of RVG of group/non-group CA, minimum number of nodes is one and maximum number of nodes is four.*

Group CA RVG follows the even distribution of RMTs which are enlisted in Table 6. So, root node (0th level) generates two nodes, each containing four RMT values at 1st level. Each of these two nodes generates another two nodes for next level (2nd level) of RVG. So, four nodes are generated using evenly distributed eight (0–7) RMT values. Each node consists of four RMT values and these RMT values have been generated using two RMTs of the previous level. Thus, all the eight RMT values are required to generate four nodes (each containing four RMT values) for the next level of RVG. For all the next levels of RVG, the total number of RMTs is same. So, more than four nodes can not be generated at any level of a group CA.

In case of non-group CA, there is some mixture of even and un-even distribution of RMTs. Even distribution means it follows Table 6 with the logic of maximum four node generation at any level. Un-even distribution means it follows Table 8. From the combination of RMTs in Table 8, maximum three nodes can be generated, which has been proved in Theorem 4.7.

To start a RVG, a root node is minimum required and to stop a RVG, a leaf node is minimum required. Similarly, minimum a node is required at each level of RVG for maintaining the connection between root node and leaf node through each cell of the CA.

Hence, the proof.

SMACA conditions for a RVG are as follows:

1 If a RVG consists of (0–7) node at any level except root and leaf node, then the RVG corresponds to a SMACA. (sufficient condition)

2 Uneven distribution of RMTs on 0-edge and/or 1-edge. (necessary condition)

3 SMACA RVG can not generate more than three nodes at any intermediate level. (necessary condition)

Theorem 4.5 *If a RVG consists of (0–7) node at any intermediate level, then the RVG corresponds to a SMACA. This is a sufficient condition for SMACA.*

In Figure 4, it is shown that a SMACA RVG consists of (0–7) RMTs in a single node. That means some other states of the corresponding CA falls on a particular attractor state directly/indirectly to form the attractor basin. So, we can say that a RVG, consisting of (0–7) RMTs within a single node of any intermediate level, generates SMACA. On the other hand, we have used the next state transition behaviour on the RV <2, 44, 4, 21> using RMT table. The same result of state distribution has been found in both the procedures. In this 4-cell CA, the only SLA is '0001'. From this discussion, we can

conclude that (0–7) RMTs within a single node of RVG makes the CA, a SMACA. It is a necessary condition.

In Figure 5, we have shown another example of RVG. Here, the RV is <13, 110, 237, 201, 81>. In this 5-cell CA, there is not a single node consisting of (0–7) RMTs at any intermediate level. So, we can say that it is not a SMACA. But, after using state transition of all the states of this 5-cell CA, we have seen that the RV <13, 110, 237, 201, 81> is also structured like a SMACA containing SLAs (‘01111 and ‘10011’). So, at this stage, we can not say that a SMACA must have (0–7) RMTs in its RVG. Though, we can say that a RVG, containing a node having value (0–7) RMTs, is a SMACA.

Based on this overall discussion, we conclude that the above mentioned condition for SMACA is a sufficient condition but not a necessary condition.

Theorem 4.6 *Un-even distribution of RMTs on 0/1 edges. This is a necessary condition for SMACA.*

From Theorem 4.5, it is proved that a RVG is a SMACA, if it has a node containing (0–7) RMTs at any intermediate level. It is also proved that it is a sufficient condition and not a necessary condition. In both the Figure 4 and Figure 5, it has been observed that each RVG consists of some un-even distribution of RMTs. So, it is a necessary condition for SMACA. Table 8 shows all the combinations of un-even distribution of RMTs. So, a SMACA must contain any type of distribution mentioned in Table 8. But, it does not mean that these RMT distributions generate SMACA. So, it is not a sufficient condition but a necessary condition.

Theorem 4.7 *SMACA RVG can not generate more than three nodes at any intermediate level of RVG. This is a necessary condition for SMACA.*

In a RVG, a node can generate maximum of two nodes for the next level through 0-edge and 1-edge. So, the root node (starting node) of RVG can generate maximum of two nodes for the next level.

Case 1 If all the RMTs (0, 1, 2, 3) fall either on 0-edge or 1-edge, it generates a single node of (0–7) RMT values for the next level.

Case 2 If the RMTs are distributed in an un-even fashion (an edge consists of one RMT value and the other edge consists of three RMT values), then the edge containing one RMT value generates a node containing two RMT values and the edge containing three RMT values generates a node containing six RMT values for the next level (1st level) of RVG. For the 2nd level of RVG, the node containing six RMT values generates maximum of two nodes and the node containing two RMT values generates either one node of four RMT values or two nodes, each containing two RMT values within which minimum one node is merged with another node (already consisting of that particular RMT value as a set/sub-set). So, for 2nd level of RVG, maximum three nodes can be generated. Similarly, for next levels of RVG, maximum three nodes are generated effectively. All other generated nodes would be merged with these three effective nodes.

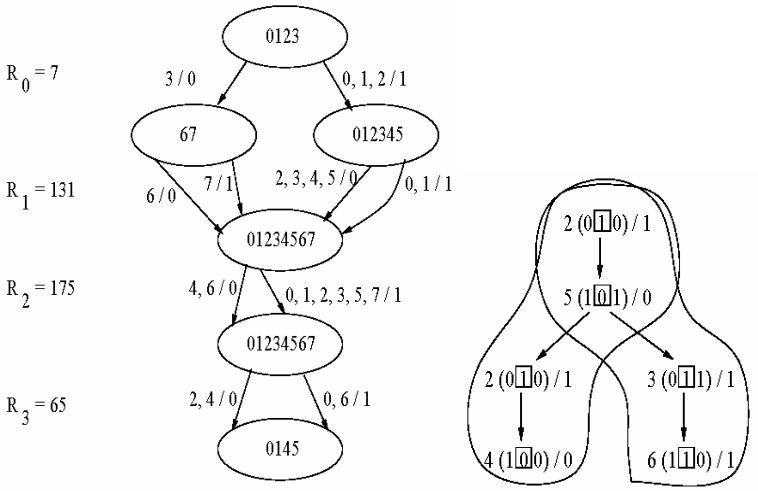
Case 3 If the RMTs are distributed in an even order (both the edges consist of two RMT values), then both the edges generate maximum of two nodes for 1st level of RVG. For the 2nd level of RVG, the RMTs of these two nodes would be

distributed either in an un-even fashion or in an even fashion. For un-even distribution, it follows 'Case 2'. But, if the distribution is even, then it will generate four nodes for the next level, which matches the general group/non-group CA condition (as per Theorem 4.4). So, it can not be a SMACA.

Thus, SMACA RVG can not generate more than three nodes at any intermediate level of RVG. So, it is a necessary condition for SMACA.

4.3 SLA detection in RVG

Figure 6 RVG and related RMT of a 4-cell CA with RV <7, 131, 175, 65>



	Rule min term (RMT)							
	111	110	101	100	011	010	001	000
7	x	x	x	x	0	1	1	1
131	1	0	0	0	0	0	1	1
175	1	0	1	0	1	1	1	1
65	x	1	x	0	x	0	x	1

Each RMT value is realised as a set of $\{a_{i-1}, a_i, a_{i+1}\}$ – 3-bit realisation of three-neighbourhood null boundary CA. Through exhaustive experimentation of state transition behaviour of a CA, we have seen that each path of a RVG corresponds to distinct states of that particular CA. Each path is a collection of edge values. An edge value consists of two parts. The left part (a_i) shows current cell value and the right part (b_i) shows the next state's cell value. If $a_i = b_i$, for every cell of a specific path of the RVG; then, the path belongs to a SLA. That means, the current state and the next state are identical. So, a self loop occurs. By left shift or right shift operation, the node values of next level are calculated from 0-edge or 1-edge values. In our experimentation, we have used only the left shift operation. In Figure 6, '3/0' is an edge value at level-0. Here, '3' means '011' – 3 bit representation of ($i-1$)th, ith and ($i+1$)th cell of current state. $a_{i-1} = 0$, $a_i = 1$ and $a_{i+1} = 1$ in this case. The value of this particular cell in the next state is

'0'. So, $b_i = 0$. In this case, $a_i = b_i$ is not achieved. That means there is no chance to develop a self loop starting from this edge. Now consider the other part of level-0. It consists of '0, 1, 2/1'. If we take '0', then $a_{i-1} = 0$, $a_i = 0$, $a_{i+1} = 0$ and $b_i = 1$. So, $a_i \neq b_i$. If we take '1', then $a_{i-1} = 0$, $a_i = 0$, $a_{i+1} = 1$ and $b_i = 1$. So again, $a_i \neq b_i$. If we take '2', then $a_{i-1} = 0$, $a_i = 1$, $a_{i+1} = 0$ and $b_i = 1$. Here, we get $a_i = b_i$. So, there is a chance to develop a SLA starting with a value $a_i = 1$. Now, using left shift operation, we get the probable two values for the next level (level-1) node. The output values are 4 and 5 as per Table 7 is concerned. For both of these values (4 and 5), the $a_i = 0$ and also $b_i = 0$. Therefore, $a_i = b_i$. There is a probability for SLA using 4 and 5. As per Table 7 is concerned, the next RMT values are 0 and 1 for 4; and, 2 and 3 for 5 at the next level (level-2). In this level-2, 0 and 1 do not satisfy $a_i = b_i$. But, 2 and 3 do satisfy $a_i = b_i$. That means, there is a chance for SLA using 2 and 3. In the next level (level-3), 2 generates 4 and 5; and 3 generates 6 and 7 as per Table 7 (left shift operation of bit values). For the last (leaf) level, 5 and 7 are ignored (right most bit value should be '0' for the right most cell of a null boundary CA). In this level, 4 and 6 both satisfy the condition $a_i = b_i$. Therefore, there is a chance for SLA using 4 and 6. The overall discussion summarises as shown in Figure 6. Finally, the SLAs are '1010' and '1011'.

5 Formation of indexing storage using SMACA

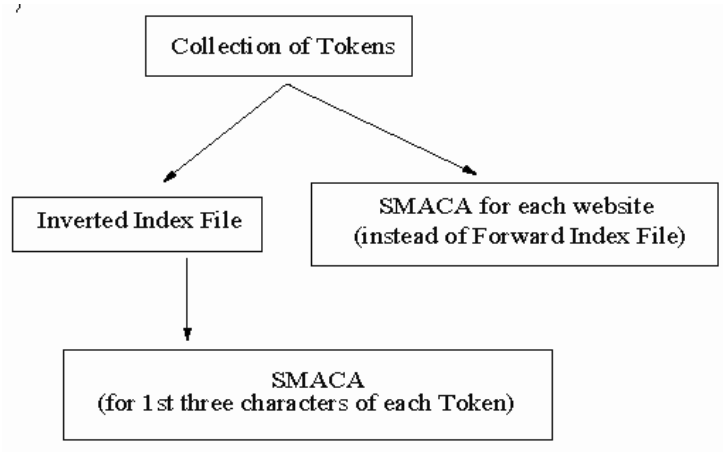
SMACA has been used as a classifier in indexing part of a search engine. SMACA based classifier can be realised in hardware in a cost effective manner since the major blocks of design is a CA having simple, regular and modular structure with local neighbourhood. Multiple CAs can be realised from a single programmable CA (PCA) that can be programmed to realise any three-neighbourhood CA. The hardware implementation of multi-class classifier has been reported in Chattopadhyay et al. (2000) while the design of a PCA cell is available in Chaudhuri et al. (1997). A lot of web-pages are stored in a structured way using this type of classifier; i.e., SMACA.

Definition 14 Token – minimum term by which one or more dictionary words can be managed while creating/modifying database of search engine. For example, 'traverse' is a token managing 'traversal', 'traverses', 'traversed', etc..

Definition 15 Key/state value – a unique number is assigned to every web-page for representing the web-pages as states of SMACA. This is known as key/state value.

Definition 16 Conflict – traversal from one state to another within a SMACA depends on its RMT (as shown in Table 1 for rule 90, 150, etc.). While generating a state of SMACA, if any mismatch happens, one or many bit position of the current state will not reach the next predefined state (0/1) as per the RMT table of concerned rule vector. This situation is known as conflict.

Figure 7 Pictorial representation of our approach



Non-linear SMACA is used (generated by Algorithm 1) for replacing forward indexing and inverted indexing. Tokens are generated in conventional manner like other web search engines.

Mainly four algorithms are used to accomplish our objective in four steps. These steps are as follows:

a generation of SMACA for each website

b generation of inverted indexed file

c replacing inverted indexed file by SMACA

d searching mechanism.

Figure 7 depicts pictorial representation of our current research work. It is clearly shown in the figure that step (a) and step (b) are done concurrently to reduce the generation time of indexing storage of a search engine.

The next four algorithms will describe our new approach step by step:

Algorithm 2: SMACA generation for forward indexing

Input A set of tokens of web-pages of a website

Output Set of SMACAs

Step 1 Generate key values of web-pages

Step 2 Assign key values of web-pages as self-loop attractors of the SMACA

Step 3 Generate key values of tokens

Step 4 Assign key values of tokens as non-reachable state, or, transient state of the SMACA

Step 5 If conflict occurs go to Step 3

Step 6 Generated SMACA

Step 7 Stop

Definition 17 Website identification number (WSID) – unique identification number has been alloted to each website. This is known as WSID.

Algorithm 3: Inverted indexed file generation

Input Token

Output Inverted indexed file

Step 1 Generate WSID

Step 2 Search for token whether it already exists in inverted indexed file

Step 3 If successful, link the WSID with the token

Step 4 Else, make a new entry in inverted indexed file and link the WSID with the token

Step 5 Stop

Definition 18 SMACA-ID – unique identification number has been allotted to each generated SMACA. This is known as SMACA-ID.

Question Why do first three characters of token take into consideration while generating SMACA-ID?

Answer After vigorous searching through WWW, it has been found that a token of any web-page consists of a minimum of three characters. Less than three character words are generally the 'stop-words'. That's why, we have taken first three characters of token into consideration for generating SMACA-ID. For example, 'sachin' and 'sacrifice' both tokens have same first three characters 'sac'. So, Algorithm 4 generates a SMACA-ID for a specific SMACA within which both the tokens reside as the states. WSIDs of the websites, within which the related tokens appear, will be assigned as attractors of that particular SMACA.

Algorithm 4: SMACA generation from inverted indexed file

Input Inverted indexed file

Output Set of SMACAs equivalent to inverted indexed file

Step 1 For each combination of first three characters of token, generate SMACA-ID from the input file

Step 2 Assign WSIDs for which first three characters of related token appear, as attractors of SMACA

Step 3 For each token matching first three characters, generate key value

Step 4 For each token, concatenate WSIDs and generated key value of token as a state value

Step 5 Assign each state value as a non-reachable or transient state of SMACA

Step 6 If conflict occurs go to Step 2

Step 7 Store generated SMACAs with corresponding SMACA-ID

Step 8 Stop

Algorithm 5: Users search

Input Users' query

Output Desired web-pages

Step 1 When query is submitted tokens are generated for the words in query

Step 2 First three characters of each token are extracted

Step 3 These three characters are encoded and SMACA-IDs are generated

Step 4 With these generated SMACA-IDs, the searcher searches the corresponding SMACA (replacing inverted indexed file) from the storage

Step 5 State values are generated from the tokens

Step 6 Applying these SMACAs with the state values, the corresponding WSIDs are found

Step 7 The searcher searches for the SMACAs (replacing forward indexing) for these WSIDs

Step 8 Applying these SMACAs with the state values previously generated by tokens, the corresponding web-page for each website (attractor) is found

Step 9 These web-pages are extracted from the repository and displayed to the user

Step 10 Stop

6 Experimental results

This section reports a detailed study on non-linear three-neighbourhood null boundary SMACA based designing on storage of hypertext data while building a web search engine. Our experiment shows that it takes less storage space and less time while searching through network. For experimental purpose, we have considered a huge number of websites within which we have shown the details of eight websites and only four web-pages of each website as a sample study within this paper.

List of websites and corresponding web-pages with details are given below:

1 AceWebTech

http://www.acewebtech.com/index.htm (35 bytes) (No. of tokens = 129)

http://www.acewebtech.com/webservices/website maintenance.htm (62 bytes) (No. of tokens = 222)

http://www.acewebtech.com/pofile.htm (36 bytes) (No. of tokens = 279)

http://www.acewebtech.com/webservices/services.htm (50 bytes) (No. of tokens = 260)

To store all these four web-pages we need a CA of size = 13 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (35 + 62 + 36 + 50) bytes = 183 bytes.

Maximum space required in our approach = (13 x 3) bytes = 39 bytes.

2 AnimalSafari

http://www.animalsafari.com/index.htm (37 bytes) (No. of tokens = 183)

http://www.animalsafari.com/html/Admissions.htm (47 bytes) (No. of tokens = 241)

http://www.animalsafari.com/html/Attractions.htm (48 bytes) (No. of tokens = 3)

http://www.animalsaari.com/html/Park Lore.htm (46 bytes) (No. of tokens = 277)

To store all these four web-pages we need a CA of size = 13 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (37 + 47 + 48 + 46) bytes = 178 bytes.

Maximum space required in our approach = (13 x 3) bytes = 39 bytes.

3 CARL

http://carltig.res.in/index.html (32 bytes) (No. of tokens = 398)

http://carltig.res.in/pages/group.html (38 bytes) (No. of tokens = 64)

http://carltig.res.in/pages/publication.html (44 bytes) (No. of tokens = 980)

http://carltig.res.in/pages/research.html (41 bytes) (No. of tokens = 155)

To store all these four web-pages we need a CA of size = 14 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (32 + 38 + 44 + 41) bytes = 155 bytes.

Maximum space required in our approach = (14 x 3) bytes = 42 bytes.

4 Domain

http://www.domain.com/ (22 bytes) (No. of tokens = 100)

http://www.domain.com/about/ (28 bytes) (No. of tokens = 198)

https://secure.registerapi.com/KM/index.php?PHPSESSID=19e124d 8a2ba2

86510720476a302087b (86 bytes) (No. of tokens = 74)

https://secure.registerapi.com/services/whois.php?siteid=42566 (62 bytes) (No. of tokens = 119)

To store all these four web-pages we need a CA of size = 12 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (22 + 28 + 86 + 62) bytes = 198 bytes.

Maximum space required in our approach = (12 x 3) bytes = 36 bytes.

5 ESPN

http://espnstar.com/(20 bytes) (No. of tokens = 302)

http://espnstar.com/cricket/cricket index.html (46 bytes) (No. of tokens = 349)

http://espnstar.com/football/football index.html (48 bytes) (No. of tokens = 475)

http://espnstar.com/cricket/cricket news.html (45 bytes) (No. of tokens = 440)

To store all these four web-pages we need a CA of size = 14 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (20 + 46 + 48 + 45) bytes = 159 bytes.

Maximum space required in our approach = (14 x 3) bytes = 42 bytes.

6 Maps of India

http://www.mapsofindia.com/outsourcing-to-india/history-of-outsourcing.html (75 bytes) (No. of tokens = 498)

http://www.mapsofindia.com/reference-maps/geography.html (56 bytes) (No. of tokens = 302)

http://www.mapsofindia.com/maps/india/india.html (48 bytes) (No. of tokens = 2388)

http://www.mapsofindia.com/stateprofiles/index.html (51 bytes) (No. of tokens = 259)

To store all these 4 web-pages we need a CA of size = 15 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (75 + 56 + 48 + 51) bytes = 230 bytes.

Maximum space required in our approach = (15 x 3) bytes = 45 bytes.

7 Tourism of India

http://www.tourism-of-india.com/adventure-tours-to-india.html (61 bytes) (No. of tokens = 133)

http://www.tourism-of-india.com/festival-tours-of-india.html (60 bytes) (No. of tokens = 158)

http://www.tourism-of-india.com/historical-places-in-india.html (63 bytes) (No. of tokens = 525)

http://www.tourism-of-india.com/kolkata.html (44 bytes) (No. of tokens = 585)

To store all these four web-pages we need a CA of size = 14 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (61 + 60 + 63 + 44) bytes = 228 bytes.

Maximum space required in our approach = (14 x 3) bytes = 42 bytes.

8 Wolfram research

http://scienceworld.wolfram.com/astronomy/(42 bytes) (No. of tokens = 39)

http://scienceworld.wolfram.com/chemistry/(42 bytes) (No. of tokens = 41)

http://mathworld.wolfram.com/(29 bytes) (No. of tokens = 234)

http://scienceworld.wolfram.com/physics/(40 bytes) (No. of tokens = 40)

To store all these four web-pages we need a CA of size = 12 with four (4) number of attractors.

In forward indexing:

Total space required in conventional way = (42+42+29+40) bytes = 153 bytes.

Maximum space required in our approach = (12x3) bytes = 36 bytes.

In inverted indexing:

Total space required for all the eight websites in conventional way = 71594 bytes.

Maximum space required for all the eight websites in our approach = 47530 bytes.

In inverted indexing,

$$SSR = (SRCA - SROA)/SRCA$$
$$= (71594 - 47530)/71594$$
$$= 0.34$$

where, *SSR* = Space saving ratio

SRCA = Space required for conventional approach

SROA = Space required for our approach

The space required for forward indexing and inverted indexing are shown in Figure 8 and Figure 9 respectively. The time required for searching is shown in Table 9 with some examples.

Figure 8 Space required for forward indexing

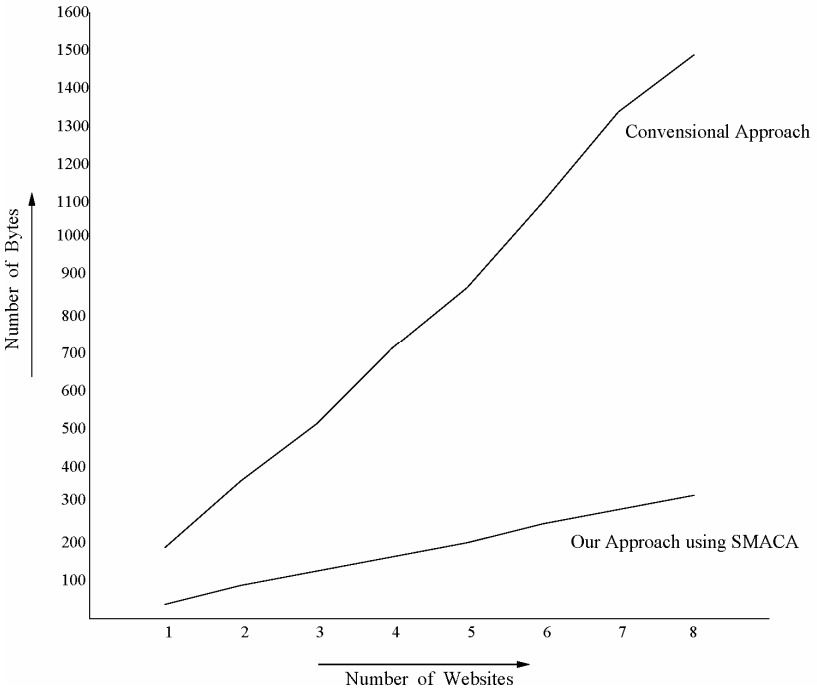


Figure 9 Space required for inverted indexing

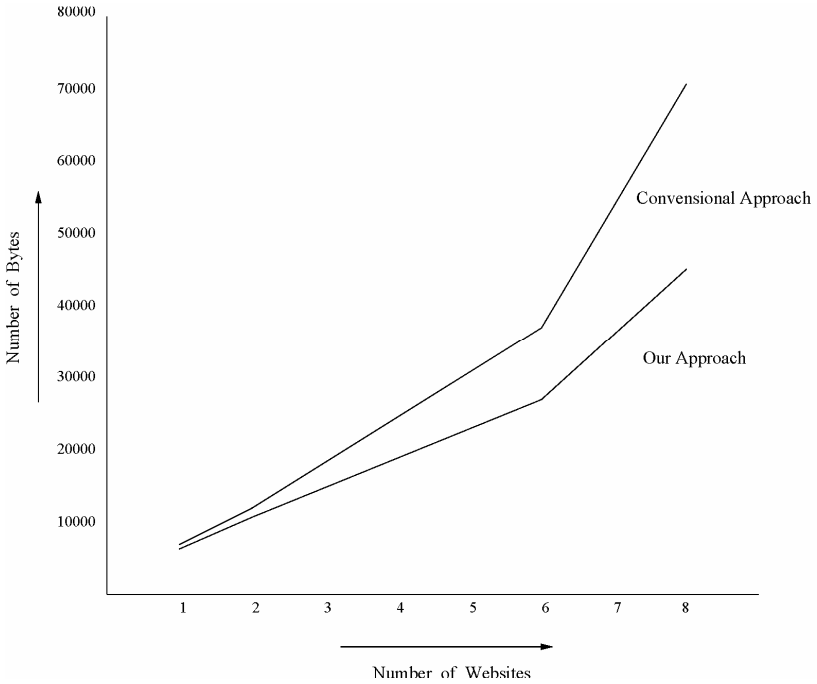


Table 9 Experimental results on time required for searching

Search samples	*No. of website (time in seconds)*			
	1	*2*	*6*	*8*
Ace	0.016	0.016	0.016	0.019
Reliable	0.016	0.016	0.018	0.020
Forum	0.016	0.018	0.019	0.021
Flash	0.016	0.016	0.017	0.018
Ace + reliable	0.016	0.016	0.018	0.019
Ace + reliable + forum	0.016	0.016	0.018	0.018
Ace + reliable + forum + flash	0.016	0.016	0.018	0.018
Hyena		0.018	0.025	0.037
Encyclopedia		0.024	0.024	0.036
Unfortunately		0.023	0.024	0.038
Mancaus		0.016	0.016	0.021
Hyena + encyclopedia		0.024	0.024	0.035
Hyena + encyclopedia + unfortunately		0.023	0.024	0.038
Hyena + encyclopedia + unfortunately + mancaus		0.023	0.024	0.038
Hyena + encyclopedia + unfortunately + mancaus + ace		0.025	0.024	0.040
Sahyadri			0.028	0.040
Kanniyakumari			0.036	0.053
Mahendra			0.033	0.051
Peninsular			0.030	0.032
Wolfram				0.039
BSI				0.058
Choice				0.018
Encyclopedia + kanniyakumari				0.039
Encyclopedia + kanniyakumari + wolfram				0.039

7 Conclusions

In a general search engine, forward indexing and inverted indexing files are used for searching. A new methodology is discussed here to minimise storage requirement by using non-linear SMACA while building forward and/or inverted indexed file. This approach processes users' query in linear time complexity while searching the web through a search engine. Using CA in storing search engine indexing data is a tricky approach that has been successfully implemented in this work offering better results in form of space efficiency.

Acknowledgements

The authors would like to acknowledge the many helpful suggestions of the participants of 9th International Conference on Parallel Computing Technologies (PaCT-2007), Pereslavl-Zalessky, Russia on earlier version of this paper.

References

Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A. and Raghavan, S. (2001) 'Searching the web', *ACM Transactions on Internet Technology*, August, Vol. 1, No.1.

Brin, S. and Page, L. (1998) 'The anatomy of a large-scale hypertextual web search engine', *Proceedings of the Seventh International World Wide Web Conference*, April, Brisbane, Australia.

Chakrabarti, S., Dom, B.E., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D. and Kleinberg, J. (1999) 'Mining the web's link structure', *IEEE Computer*, August, Vol. 32, No. 8, pp.60–67.

Chattopadhyay, S., Adhikari, S., Sengupta, S. and Pal, M. (2000) 'Highly regular, modular and cascadable design of cellular automata-based pattern classifier', *IEEE Trans. on VLSI Systems*, December, Vol. 8, No. 6.

Chaudhuri, P.P., Chowdhury, D.R., Nandi, S. and Chatterjee, S. (1997) 'Additive cellular automata, theory and applications, Vol. 1', *IEEE Computer Society Press*, Los Alamitos, California, Vol. ISBN-0-8186-7717-1.

Das, S., Kundu, A., Sen, S., Sikdar, B.K. and Chaudhuri, P.P. (2003) 'Non-linear cellular automata based PRPG design (without prohibited pattern set) in linear time complexity', *Asian Test Symposium*, pp.78–83.

Das, S., Kundu A. and Sikdar, B.K. (2004) 'Non-linear CA based design of test set generator targeting pseudo-random pattern resistant faults', *Asian Test Symposium*, pp.196–201.

Flake, G.W., Lawrence, S., Giles, C.L. and Coetzee, F.M. (2000) 'Self organization and identification of web communities', *IEEE Computer*, Vol. 35, No. 3, pp.66–71.

Glover, E.J., Tsioutsiouliklis, K., Lawrence, S., Pennock, D.M. and Flake, G.W. (2002) 'Using web structure for classifying and describing web pages', *WWW2002*, Honolulu, Hawaii, USA, May, pp.7–11.

Kundu, A., Dutta, R. and Mukhopadhyay, D. (2007) 'Generation of SMACA and its application in web services', *9th International Conference on Parallel Computing Technologies, PaCT 2007 Proceedings*, Pereslavl-Zalessky, Russia, Lecture Notes in Computer Science, Springer-Verlag, Germany, September 3–7.

Maji, P., Shaw, C., Ganguly, N., Sikdar, B.K. and Chaudhuri, P.P. (2003) 'Theory and application of cellular automata for pattern classification', *Fundamenta Informaticae*, December, Vol. 58, pp.321–354.

Mukhopadhyay, D. and Singh, S.R. (2004) 'An algorithm for automatic web-page clustering using link structures', *Proceedings of the IEEE INDICON 2004 Conference*, IIT Kharagpur, India, 20–22 December, pp.472–477.

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

Sipper, M. (1996) 'Co-evolving non-uniform cellular automata to perform computations', *Physica D*, Vol. 92, pp.193–208.

Wolfram, S. (1986) 'Theory and application of cellular automata', *World Scientific*.