Web toolkit: an agent based scalable search engine using cellular automata based classification for duly ranked retrieved data

Anirban Kundu*

Netaji Subhash Engineering College,
West Bengal University of Technology,
West Bengal-700 152, India
and
Web Intelligence and Distributed Computing Research Lab,
(WIDiCoReL), Green Tower C-9/1, Golf Green,
Calcutta 700095, India
E-mail: anik76in@gmail.com
*Corresponding author

Debajyoti Mukhopadhyay

Calcutta Business School,
Diamond Harbour Road,
Bishnupur, West Bengal-743503, India
and
Web Intelligence and Distributed Computing Research Lab,
(WIDiCoReL), Green Tower C-9/1, Golf Green,
Calcutta 700095, India
E-mail: debajyoti.mukhopadhyay@gmail.com

Abstract: Web page classification is a major issue for categorising web documents to facilitate indexing, search and retrieval of web pages for search engine. Different crawling techniques have been utilised to accumulate web pages of different domains under separate databases depending on practical scenario. Downloaded web pages are being parsed for further processing. A classifier is designed dynamically using single cycle multiple attractor cellular automata for mapping downloaded web pages of different domains into specific structure. This paper proposes alternate technique for automatic categorisation of web pages into different domains. Retrieved web pages have been ranked automatically at the time of classifier formation. Typically, our system consists of crawling, ranking and storage parts created in a different way. Hierarchical concept has been used over parallel crawler. GF(2P) concept is introduced in ranking. The concept of SMACA has been utilised in indexing storage. Overall, a search engine module has been created using agent-based method.

Keywords: cellular automata; CA; Galois field; GF; hierarchical crawler; page rank; single cycle multiple attractor cellular automata; SMACA.

Reference to this paper should be made as follows: Kundu, A. and Mukhopadhyay, D. (2011) 'Web toolkit: an agent based scalable search engine using cellular automata based classification for duly ranked retrieved data', *Int. J. Intelligent Information and Database Systems*, Vol. 5, No. 2, pp.177–199.

Biographical notes: Anirban Kundu is an Assistant Professor in Information Technology Department of Netaji Subhash Engineering College, India. He is also a Research Fellow in the Web Intelligence and Distributed Computing Research Lab (WIDiCoReL). He received his BE Mechanical from Islamiah Institute of Technology, Bangalore University (1999). He also received his Post Graduate Diploma in Financial Management from Management Studies Promotion Institute (2001), an MTech (IT) from Bengal Engineering and Science University (2004), and PhD (Engineering) in Computer Science from Jadavpur University (2009). His research interests include search engine oriented indexing, ranking, prediction, web page classification, semantic web (ontology-based) and natural language processor with the essence of cellular automata. He is also interested in multi-agent-based system design, fuzzy controlled systems and cloud computing. He has presented his research work in England, Russia and India at various international conferences.

Debajyoti Mukhopadhyay is the Founder Director of Web Intelligence & Distributed Computing Research Lab (WIDiCoReL), and is a Professor and the Head of Information Technology & MIS at Calcutta Business School. He was earlier with Bell Communications Research, USA. He was a full Professor of Computer Science and Engineering at the West Bengal University of Technology affiliated Engineering Colleges (2001–2008). He was a Visiting Professor in the Division of Electronics and Information Engineering at Chonbuk National University, Korea (2006–2007). He had also taught at Stevens Institute of Technology, USA (1982–1984) and at Bengal Engineering College, Calcutta University (1980–1981). His research areas are in: web data mining and distributed systems. He received his BE (Electronics) from Calcutta University, DCS from The Queen's University of Belfast, MS from Stevens Institute of Technology and PhD (Computer Science) from Jadavpur University, India.

1 Introduction

In present day situation, internet represents all fields of our society. Information on the web is authored and organised by billions of different people. Thus, searching World Wide Web (WWW) considered as a library having widely distributed information posses a great challenge (Arasu et al., 2001).

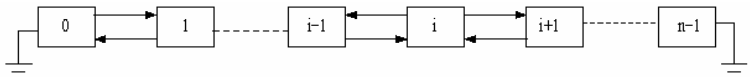
Web page classification is a necessity to fulfil our daily needs in web-based culture. Every day, lots of newly created web pages are being added in the WWW. A large variety of noisy information is embedded into web pages. That's why the classification of web pages is pretty complicated. Using our proposed technique, a large amount of information can be categorised and further ranked in a particular fashion with minimum time complexity (Brin and Page, 1998).

A classifier has various applications in many fields of science and society. Classifier can be viewed as decision tree, which accepts values of some features or characteristics of a situation as input and produces discrete output. We have divided our proposed work into two phases. First phase consists of crawling for topic-based web pages through single, parallel and hierarchical crawling methods. Then, web pages are saved and further classified using SMACA in second phase. The classifier contains web pages at different levels as per their ranking value. At the time of users' query, the desired results are

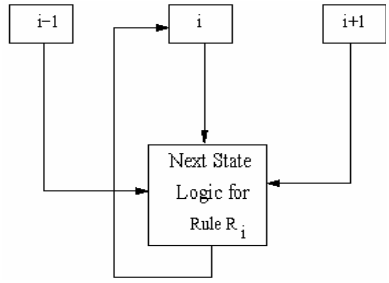
displayed as already stored in the classifier (Risvik and Michelsen, 2002; Kobayashi and Takeda, 2000).

In Section 2, we present cellular automata basics. Then, we present web page classification concept in Section 3. In Section 4, our approach is described along with algorithms. In Section 5, experimental results are shown. Finally, we conclude our work in Section 6.

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

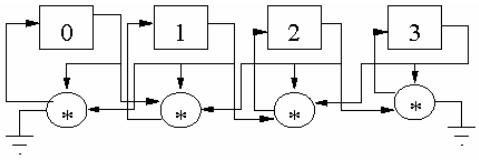


(a)

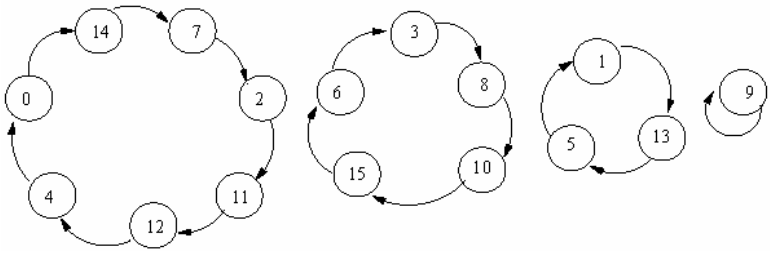


(b)

Figure 2 State transition behaviour of cellular automata rule vector <6 240 60 65>, (a) CA structure (b) state transition behaviour displaying four cycles of length 7, 5, 3 and 1



(a)



(b)

Note: *denotes the next state logic corresponding to rule R_i (i = 0, 1, 2, 3) in the rule vector $<R_0\ R_1\ R_2\ R_3>$ for *i*th cell.

2 Cellular automata basics

An *n* cell CA consists of *n* cells [Figure 1(a)] with local interactions. 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 (three bit binary string) represents a minterm of a three variable Boolean function for a three neighbourhood CA cell. Figure 2 shows the structure [Figure 2(a)] and state transition behaviour [Figure 2(b)] of a CA having the rule vector as <6 240 60 65>.

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 – the eight minterms having decimal values 0, 1, 2, 3, 4, 5, 6, 7 are referred to as rule minterms (RMTs)

Present states of 3-neighbours ($i-1$), i, and ($i+1$) of ith cells (minterms of a three variable Boolean function)	111 (7) T(7)	110 (6) T(6)	101 (5) T(5)	100 (4) T(4)	011 (3) T(3)	010 (2) T(2)	001 (1) T(1)	000 (0) T(0)	Rule number
Next state of *i*th cell	0	1	0	1	1	0	1	0	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: Set of minterms $T = \{7, 6, 5, 4, 3, 2, 1, 0\}$ represented as {T(7), T(6), T(5), T(4), T(3), T(2), T(1), T(0)} (T(m) = m, m = 0 to 7) in the text, are noted simply as q.

In subsequent discussions, each of the eight entries in Table 1 is referred to as a rule minterm (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 eight bit combination is referred to as a CA rule (Wolfram, 1986). The rule of a CA cell represents its next state logic as illustrated in Table 2 for a few example rules. It can be derived from the truth table (Table 1) of the *i*th cell, where q_i^{t+1} is the next state of *i*th cell, while q_{i-1}^t, q_i^t, and q_{i+1}^t are the current states of (i – 1)th, *i*th, and (i + 1)th cells respectively; the $\oplus$ represents XOR logic and ($\cdot$) denotes AND function. If a CA cell is configured with a specific rule, its next state function implements the truth table as illustrated for sample rules in Table 2. The first two rules 90 and 150 of Table 2 are linear rules employing XOR logic while remaining non-linear rules employ AND logic in addition to XOR. Out of 256 possible rules, as shown in Table 3, there are seven rules with XOR logic and another seven rules employ XNOR logic. The rule 0 sets the cell to state '0' for each of the eight minterms. The remaining rules are non-linear rules employing AND/OR/NOT logic. Linear and additive CA employing XOR/XNOR logic have been characterised with matrix algebraic formulation (Chaudhuri et al., 1997).

Table 2 Next state logic of a few rules

Rule 90	$q_i^{t+1} = q_{i-1}^t \oplus q_{i+1}^t$
Rule 150	$q_i^{t+1} = q_{i-1}^t \oplus q_i^t \oplus q_{i+1}^t$
Rule 120	$q_i^{t+1} = q_{i-1}^t \oplus \left(q_i^t \cdot q_{i+1}^t\right)$
Rule 12	$q_i^{t+1} = q_{i-1}^t \oplus \left(q_{i-1}^t \cdot q_i^t\right)$
Rule 210	$q_i^{t+1} = q_{i-1}^t \oplus q_{i+1}^t \oplus \left(q_i^t \cdot q_{i+1}^t\right)$

Notes: $\oplus$ represents XOR while $(\cdot)$ denotes AND logic functions.

Table 3 Linear/additive CA rules employing next state function with XOR/XNOR logic

Rule no.	*Next state function with XOR logic*	*Rule no.*	*Next state function with XNOR logic*
Rule 60	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t)$	Rule 195	$\overline{q_i(t+1) = q_{i-1}(t) \oplus q_i(t)}$
Rule 90	$q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$	Rule 165	$\overline{q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)}$
Rule 102	$q_i(t+1) = q_i(t) \oplus q_{i+1}(t)$	Rule 153	$\overline{q_i(t+1) = q_i(t) \oplus q_{i+1}(t)}$
Rule 150	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$	Rule 105	$\overline{q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)}$
Rule 170	$q_i(t+1) = q_{i+1}(t)$	Rule 85	$\overline{q_i(t+1) = q_{i+1}(t)}$
Rule 204	$q_i(t+1) = q_i(t)$	Rule 51	$\overline{q_i(t+1) = q_i(t)}$
Rule 240	$q_i(t+1) = q_{i-1}(t)$	Rule 15	$\overline{q_i(t+1) = q_{i-1}(t)}$

Note: Rule 0 sets the cell to state '0' for each of the eight minterms.

2.1 Definitions

The following definitions are highly required to understand the basics of CA. In this paper, CA is utilised as classifier in construction of search engine storage.

Definition 2.1: 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 2.2: Non-group CA – A non-group CA has states that have r number of predecessors, where $r = 0, 1, 2, 3, \cdots$.

Definition 2.3: Reachable state – A state having 1 or more predecessors is a reachable state.

Definition 2.4: Non-reachable state – A state having no predecessor (i.e., r = 0) is termed as non-reachable.

Definition 2.5: Cyclic state – A state in a cycle of the state transition behaviour of a CA. Cyclic states are also referred to as Stable (semi stable) states.

Definition 2.6: Transient state – A non-cyclic state of a non-group CA is referred to as a transient state.

Definition 2.7: Attractor cycle – The set of states in a cycle is referred to as an attractor cycle.

Definition 2.8: Self-loop attractor (SLA) – A single cycle attractor state with self-loop is referred to as SLA.

Definition 2.9: Rule vector (RV) – The sequence of rules $< R_0R_1 \cdots R_i \cdots R_{n-1}>$, where ith cell is configured with rule R_i.

3 Web page classification concept

Before going into the details of classification concept, we should know some basic terminologies which are used in our approach. The relevance of web page depends on several factors as mentioned below:

Definition 3.1: In-link – In-link of a web page means number of incoming connections through hyperlink of other web pages.

Definition 3.2: Out-link – Out-link of a web page means number of outgoing connections through hyperlink of current web pages.

Definition 3.3: Hub-page – A web page consisting of outgoing links is known as hub-page.

Definition 3.4: Authority-page – A web page having incoming connections is known as authority-page.

Definition 3.5: Page rank – Page rank is a set of algorithms for assigning numerical weightings to web pages indexed by a search engine.

Definition 3.6: Seed queue – A queue, containing number of unified resource locators (URLs) as seed for downloading the web pages from internet, is known as seed queue.

Definition 3.7: Valid topic – A web page contains valid topic, whenever the contents are related to the specified topic of designing a classifier.

Definition 3.8: Single crawler – A single crawler is a set of programs by which web pages are being surfed by a search engine to download selected web pages.

Definition 3.9: Focused single crawler – Single crawler being used for a specific topic is known as focused single crawler.

Definition 3.10: Parallel crawler – A set of crawlers working concurrently is known as parallel crawler.

Definition 3.11: Focused parallel crawler – Parallel crawler being used for a specific set of topics is known as focused parallel crawler.

Definition 3.12: Hierarchical crawler – It is a set of dynamic crawlers which is created based on the number of links required to be downloaded from WWW at any level of downloading.

Definition 3.13: Focused hierarchical crawler – Hierarchical crawler being used for a specific topic is known as focused hierarchical crawler.

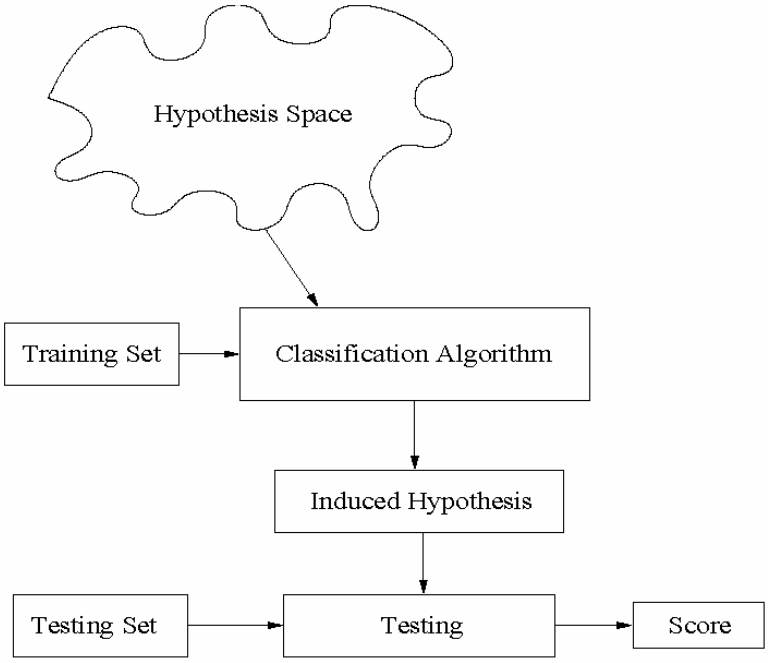
Web page classification is an important factor for search engines (Kwok, 1998; Glover et al., 2002). Search engine needs classification very badly due to huge amount of data handling. A large amount of time is saved while executing using advanced classification techniques available in the market (Mukhopadhyay and Biswas, 2005; Mukhopadhyay et al., 2003). Web page classification is one of the essential techniques for web mining since classifying web pages of any specific class is often the first step of mining the web (Chakrabarti et al., 1999; Davison, 2000). A classifier construction requires huge amount of dataset for the training phase (Wong and Fu, 2000; Attardi et al., 1999). Web page classification is much more difficult than pure text classification due to a large variety of noisy information embedded in web pages. Figure 3 shows a general diagram of an existing classifier (Flake et al., 2000).

There are different types of categorisation as well as classification.

- Manual categorisation: The traditional manual approach of classification involved the analysis of the contents of the web pages by a number of domain experts and the classification was based on the textual content. Any well maintained web page containing several category wise fields is the trivial example of this category (http://www.yahoo.com).
- Clustering approach: Clustering algorithms have been used widely as the clusters can be formed directly without any background information (Wong and Fu, 2000).
- META-tag-based categorisation: This technique relies solely on attributes of the meta tags.
- Text-content-based categorisation: To classify a document, all the stop-words are removed and the remaining keywords or phrases are represented as a vector. This document is then classified into an appropriate category (Wong and Fu, 2000).
- Link and content analysis: This approach is an automatic web page categorisation technique based on the fact that a web page that refers to a document must contain enough hints about its content to induce someone to read it. Such hints can be used to classify the document (Attardi et al., 1999).

Figure 3 is a general representation of an existing classifier which shows the various components as block of information. Initially, a known training set is required to build a classifier using selected algorithms. As per the diagram, an induced hypothesis is generated from the original hypothesis space. The ultimate score will be assessed using testing set data.

Figure 3 Schematic diagram of existing classifier



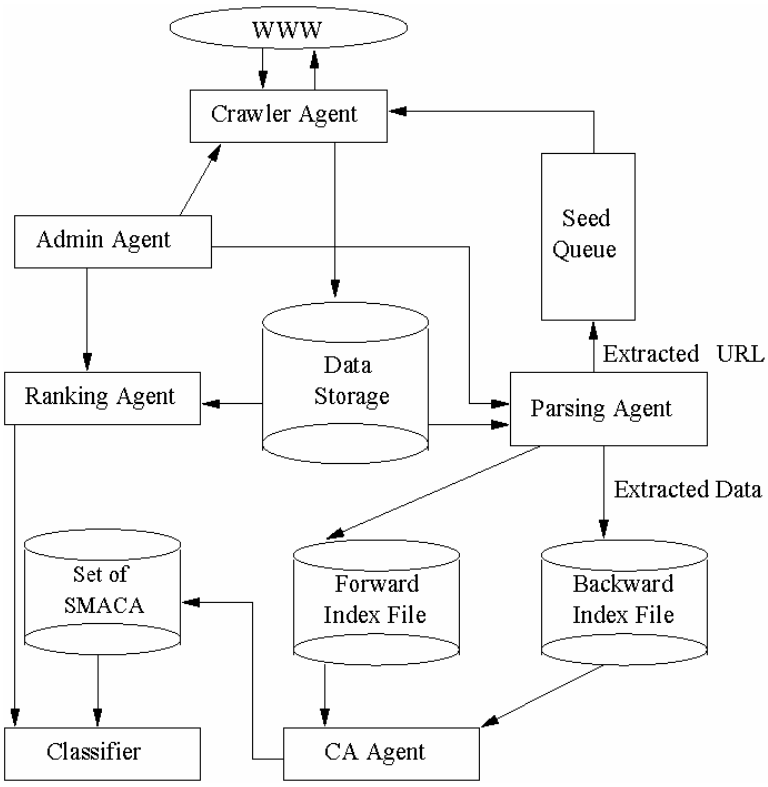
4 Our approach

In this project, our approach is to build a classifier model using CA concept. We have divided our work into two phases using agent-based modules.

Admin agent works like a backbone of the whole system. Whenever, there is a new web page to be downloaded, it invokes the crawler agent to do the specified task. Admin agent simultaneously ignites the parsing agent to parse or extract the words (except stop-words) of downloaded document for modifying the forward and inverted indexed files as well (Brin and Page, 1998). These forward and inverted indexed files are then used by CA agent to generate a set of SMACAs which contain several web pages of distinct topics. Ranking agent ranks those web pages based on specific criteria and fix the web pages at different levels of SMACAs depending on their ranking. Thus, admin agent behaves similar to a controller unit which is responsible for synchronisation of 'Phase 1' and 'Phase 2' as shown in Figure 4.

- Phase 1: web-based crawler module as discussed in Kundu et al. (2006b, 2009, 2008b)
- Phase 2: classifier module based on Mukhopadhyay et al. (2006) and Kundu et al. (2007b, 2008a).

Figure 4 Proposed structure of the agent-based classifier model



4.1 Phase 1

In the first phase of our approach, we have designed a crawler-based system to download unique web pages as per requirement based on domain knowledge. Crawler typically downloads the web pages to store into repository or storage. Then, a checking module is used for checking the downloaded web pages whether these are related to the specific domain or not.

In crawler agent, we have maintained three separate channels for three types of crawling which are known as single crawling, parallel crawling and hierarchical crawling respectively. This crawler agent generally invokes any one type of crawling sub-system based on the situation. Single crawling and parallel crawling are already known to everybody (Pinkerton, 1994; Boldi et al., 2002). Algorithm 1 describes single crawling in a brief. There is one check point which checks whether the web page is already downloaded. The web page which is not yet downloaded would be saved in the storage. Parsing agent takes the saved web page to parse the web links (hyperlinks) for further downloading if required (refer to Figure 5).

Algorithm 1 Single crawling technique

Input: A set of seed URLs within seed queue	
Output: Storage of downloaded web pages within server irrespective of domain	
Step 1	Crawler starts crawling with seed URLs
Step 2	Crawler downloads the web page taking URL from the seed queue
Step 3	Check whether the URL of downloaded web page is already visited or not
Step 4	If visited, discard the web page and go to Step 9
Step 5	Else, save the web page
Step 6	Hyperlinks of each web page are extracted using Parser tool
Step 7	Check whether the extracted URLs are already visited or not
Step 8	Save those extracted hyperlinks, which are still not visited, within Seed Queue as well as in storage
Step 9	If (condition for continuing the downloading process satisfies)
Step 10	then go to Step 2
Step 11	Stop

Figure 5 Pictorial representation of single crawling technique

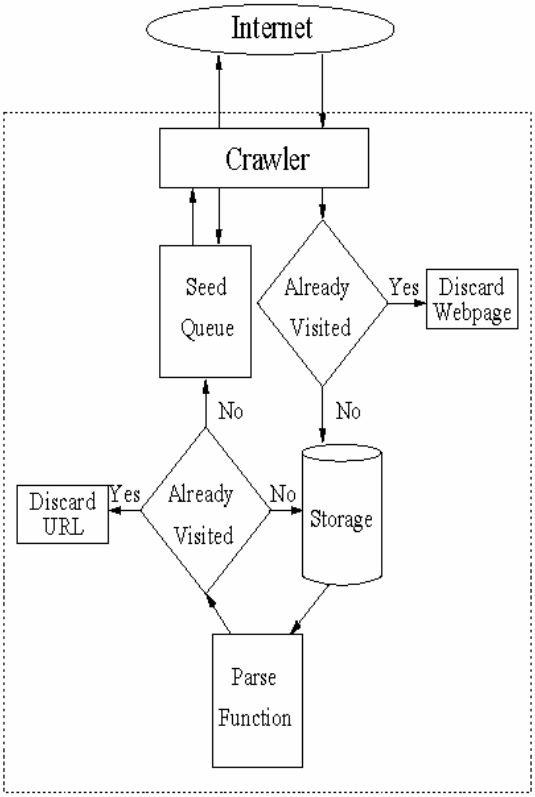
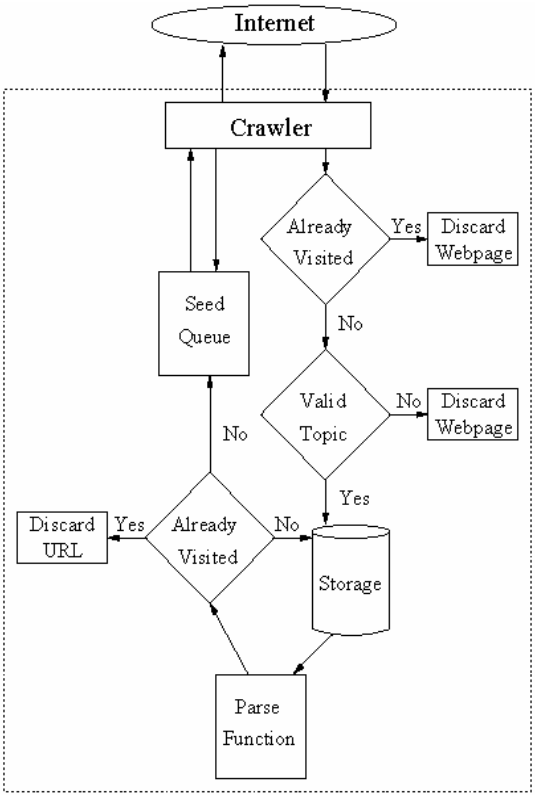


Figure 6 Pictorial representation of focused single crawling technique



Algorithm 2 Focused single crawling technique

Input: A set of seed URLs within seed queue	
Output: Storage of focused downloaded web pages within server irrespective of domain	
Step 1	Crawler starts crawling with seed URLs
Step 2	Crawler downloads the web page taking URL from the seed queue
Step 3	Check whether the URL of downloaded web page is already visited or not
Step 4	If visited, discard the web page and go to Step 11
Step 5	Else check for valid topic
Step 6	If downloaded web page does not contain valid topic, then discard the web page and go to Step 11
Step 7	Else, save the web page
Step 8	Hyperlinks of each web page are extracted using Parser tool
Step 9	Check whether the extracted URLs are already visited or not
Step 10	Save those extracted hyperlinks, which are still not visited, within seed queue as well as in storage
Step 11	If (condition for continuing the downloading process satisfies)
Step 12	then go to Step 2
Step 13	Stop

It has been found that a URL may have several synonymous addresses. To download unique web pages, we have to take care of this situation. If the downloaded web page has a URL which is already visited through crawler, then the downloaded web page would be removed from the system. For example, 'http://www.coke.com' and 'http://www.cocacola.com' both point to the same URL 'http://www.coca-cola.com/index-b.html'. Since a crawler may download web pages which are not required for specific work of interest, Focused crawling (refer to Figure 6) is further introduced using Algorithm 2 to enhance better performance. In this algorithm, another check point is being introduced. This check point is useful to check whether the downloaded web page has got the valid topic or not. Predefined database is used to check this validity of the topic.

Algorithm 3 URL transfer program

Input: Parsed URL and the corresponding downloaded web page of a particular domain	
Output: Transfer of URL to domain-based seed queue	
Step 1	If (parsed URL and input web page belong to the same domain)
Step 2	then go to Step 6
Step 3	Search for the specific domain
Step 4	If (domain found)
Step 5	then transfer the parsed URL to the specific seed queue of related domain
Step 6	Stop

Algorithm 4 Parallel crawling technique

Input: A set of seed URLs of different domains within different seed queues	
Output: Storage of downloaded web pages within servers of respective domains	
Step 1	A set of crawlers start crawling with seed URLs of their respective domains
Step 2	Each crawler downloads the web page taking URL from their respective seed queue
Step 3	Check whether the URL of downloaded web page is already visited or not
Step 4	If visited, discard the web page and go to Step 13
Step 5	Else, save the web page
Step 6	Hyperlinks of each web page are extracted using Parser tool
Step 7	Check whether the extracted URLs are already visited or not
Step 8	If not visited, then check for domain
Step 9	If extracted hyperlinks belong to same domain
Step 10	then save the hyperlinks within the respective seed queue and storage
Step 11	If extracted hyperlinks belong to different domains
Step 12	then use 'URL transfer program' to transfer the hyperlinks to their respective seed queue and storage
Step 13	If (condition for continuing the downloading process satisfies)
Step 14	then go to Step 2
Step 15	Stop

Parallel crawling part of the crawler agent offers better performance at the time of downloading the web pages from internet. Using 'N' number of crawlers, a system can download and further process the web pages 'N' times faster than single crawler and also the system overhead is much lesser since different computer machines are considered for storing web pages of different domains using Algorithm 4. Thus, whole network load is dispersed into distinct domain loads. By parallel crawling technique, web coverage is more within specific time compared to single crawling. Since parallel crawling part of the Crawler Agent uses distinct crawlers to download and store web pages of various domains, there is no chance of overlapping. A URL transfer program module is used to transfer data from one domain to another using Algorithm 3 for ease of parallel crawling (refer to Figure 7). After parsing the URLs of downloaded web pages, there is a possibility to have many URLs containing the address of other domains. So, this type of transfer module is a major requirement while designing the whole system. So, dynamic assignment is done for allocating URLs within the seed queue of distinct domains. Again, focused crawling is introduced in this scenario to penetrate for better results using Algorithm 5. The ultimate aim of a focused crawler is to selectively search for web pages that are relevant to a pre-defined set of topics. The focused crawler (refer to Figure 8) analyses its crawl limit to search the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web. This leads to significant savings in hardware and network resources.

Figure 7 Pictorial representation of parallel crawling technique

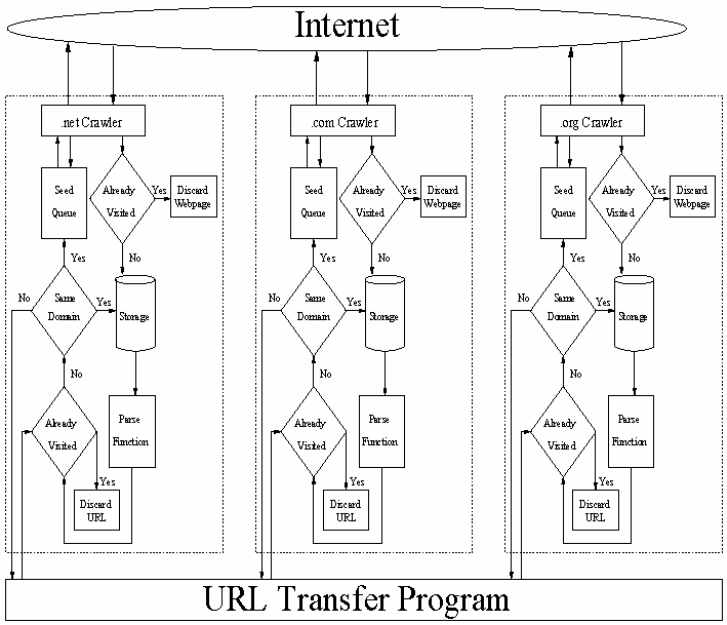
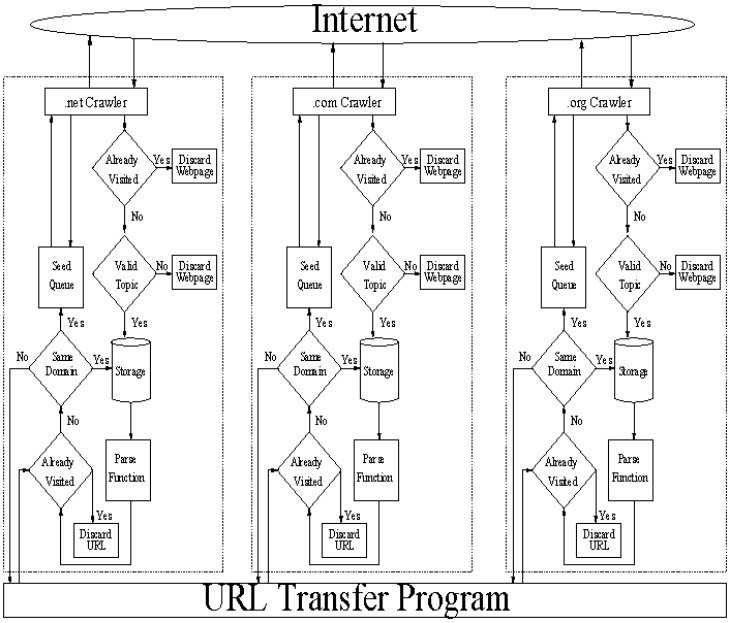


Figure 8 Pictorial representation of focused parallel crawling technique



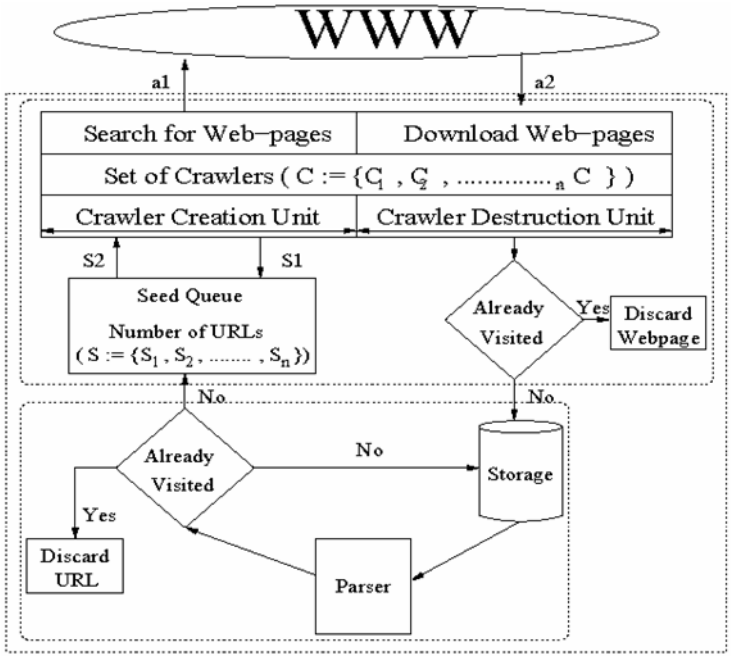
Algorithm 5 Focused parallel crawling technique

Input: A set of seed URLs of different domains within different seed queues	
Output: Storage of focused downloaded web pages within servers of respective domains	
Step 1	A set of crawlers start crawling with seed URLs of their respective domains
Step 2	Each crawler downloads the web page taking URL from their respective seed queue
Step 3	Check whether the URL of downloaded web page is already visited or not
Step 4	If visited, discard the web page and go to Step 15
Step 5	Else check for valid topic
Step 6	If downloaded web page does not contain valid topic, then discard the web page and go to Step 15
Step 7	Else, save the web page
Step 8	Hyperlinks of each web page are extracted using Parser tool
Step 9	Check whether the extracted URLs are already visited or not
Step 10	If not visited, then check for domain
Step 11	If extracted hyperlinks belong to same domain
Step 12	then save the hyperlinks within the respective seed queue and storage
Step 13	If extracted hyperlinks belong to different domains
Step 14	then use ‘URL transfer program to transfer the hyperlinks to their respective seed queue and storage
Step 15	If (condition for continuing the downloading process satisfies)
Step 16	then go to Step 2
Step 17	Stop

Algorithm 6 Hierarchical crawling technique

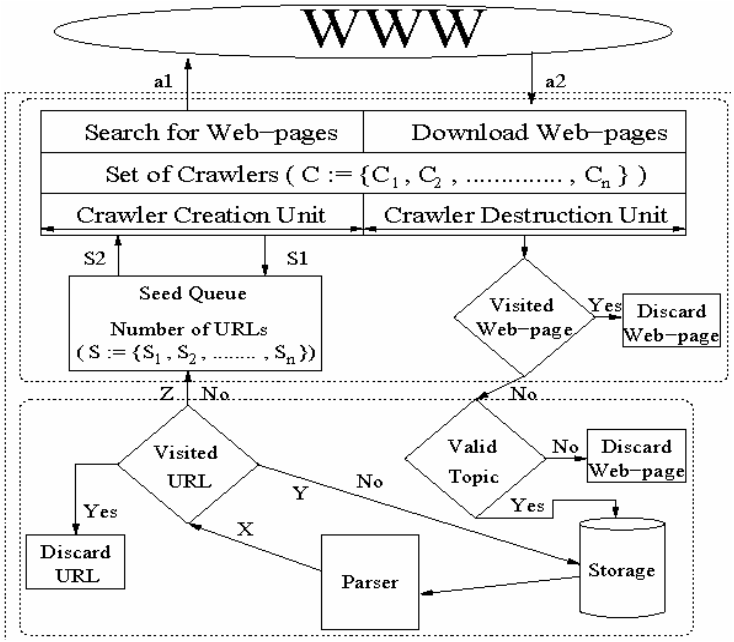
Input: A set of seed URLs within seed queue and depth level of searching	
Output: Storage of downloaded web pages	
Step 1	Initialise i with 0
Step 2	Continue loop until i > Depth
Step 3	Check number of URLs within seed queue
Step 4	Generate same number of crawlers in runtime
Step 5	Assign each seed(URL) to a specific crawler
Step 6	Download all web pages
Step 7	Kill all crawlers
Step 8	Check whether the URLs of downloaded web pages are already visited or not
Step 9	Discard already visited web pages
Step 10	Save new web pages
Step 11	Hyperlinks of all saved web pages are extracted using Parser tool
Step 12	Check whether the extracted URLs are already visited or not
Step 13	Save those extracted hyperlinks(URLs), which are still not visited, within Seed Queue as well as in storage
Step 14	Increment i by 1
Step 15	Stop

Figure 9 Pictorial representation of hierarchical crawling technique



Note: 'n' number of seed URLs are selected at a time by 'n' number of dynamically created crawlers where n = 1, 2, 3, ...

Figure 10 Pictorial representation of focused hierarchical crawling technique



Note: 'n' number of Seed URLs are selected at a time by 'n' number of dynamically created crawlers where n = 1, 2, 3, ...

Algorithm 7 Focused hierarchical crawling technique

Input: A set of seed URLs within seed queue and depth level of searching
Output: Storage of focused downloaded web pages

Step 1	Initialise i with 0
Step 2	Continue loop until i > Depth
Step 3	Check number of URLs within seed queue
Step 4	Generate same number of crawlers in runtime
Step 5	Assign each seed(URL) to a specific crawler
Step 6	Download all web pages
Step 7	Kill all crawlers
Step 8	Check whether the URLs of downloaded web pages are already visited or not
Step 9	Discard visited web pages
Step 10	New web pages are being checked for valid topic
Step 11	Discard the web pages which do not contain valid topic
Step 12	Save remaining web pages
Step 13	Hyperlinks of all saved web pages are extracted using Parser tool
Step 14	Check whether the extracted URLs are already visited or not
Step 15	Save those extracted hyperlinks, which are still not visited, within seed queue as well as in storage
Step 16	Increment i by 1
Step 17	Stop

The third part of the crawler agent is the hierarchical crawling. In this part of the crawler agent, the crawling system is further enhanced to achieve better performance with respect to time. There are some situations where it is highly required because of the uncertainty of predefined calculations on 'number of crawlers' requirement'. For example, any type of web-based tutorial downloading has different number of links at different levels. In this scenario, hierarchical crawling concept is required. It is basically the advanced form of parallel crawling where there is no fixed number of crawlers. Its nature is very dynamic. The numbers of crawlers are calculated at each level of downloading web pages from WWW. So, each level may have different number of links (URLs) which can not be realised and/or calculated in advance. Algorithm 6 describes the hierarchical crawling (refer to Figure 9). Focused crawling is introduced in hierarchical crawling also for specific topic searching. It analyses its crawl limit to search the links which are looking more relevant for crawling as mentioned in Algorithm 7 (refer to Figure 10). Follow Kundu et al. (2006b, 2009, 2008b) for detailed study on crawling mechanism.

The output of Phase I is saved into the data storage (refer to Figure 4) for further processing in Phase 2.

4.2 Phase 2

The crawlers pass the retrieved web pages into the concerned repository. There is a Parsing agent which extracts all the words or tokens from each retrieved web page, and records the URL where each word occurred in form of forward index. Forward index is sorted on document identification number (do-cID) 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) (Mukhopadhyay and Singh, 2004; Furnkranz, 1999). In the second phase of our approach, we have designed a classifier taking above mentioned downloaded web pages as input. These web pages are already focused to a pre-defined topic and domain specific. Firstly, these web pages are ranked as per relevance described in Kundu et al. (2006a, 2007a) using Algorithm 8.

Algorithm 8 Web page ranking based on X-link

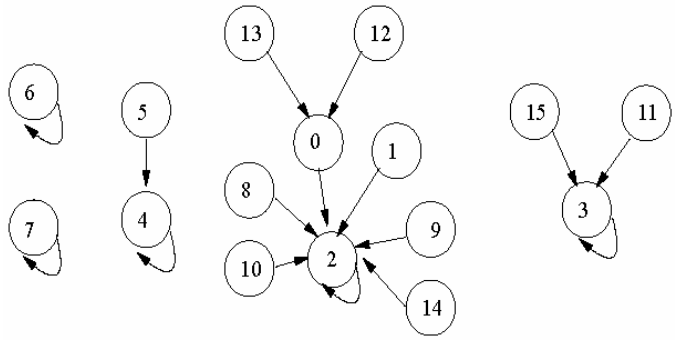
Input: Number of web pages (no wp)	
Output: Page Rank (PR) of all web pages	
Step 1	Set Connection Matrix (CMat)
Step 2	Calculate X-links of each web page
Step 3	Loop(Start)
Step 4	For i := 1 to no wp; repeat Step 5-7
Step 5	For j := 1 to no wp; repeat Step 6-7
Step 6	Sum_of_page_rank := Sum_of_page_rank + page_rank[i] / X-link[j]
Step 7	Page_Rank[i] := (1-0.85) + 0.85 * Sum_of_page_rank // 0.85 is Damping Factor
Step 8	Loop(Stop) after settle-up the rank of each web page
Step 9	Stop

Notes: Typically, for Hub, X-link = out-link and for Authority, X-link = in-link

Here, CMat is a [nxn] matrix containing '0' or '1'. Now, consider every cell of CMat as dependent (1), or, independent (0) in nature. In Table 2, it is clearly shown that next state value of Rule 90 depends on $(i - 1)$th and $(i + 1)$th cell of the current state. It means the row containing the value of Rule 90 has numeric '1' in its left and right position of the diagonal. All other cells along with the diagonal contain numeric '0'. In this way, RMT can be generated depending on the CMat. Further, CA rules are being developed from the corresponding RMT table (such as Table 1). Table 1 is the example of three-neighbourhood CA cells. In practical scenario, 'n' neighbourhood CA cells should be generated for better accuracy. Finally, generated CA rules are stored instead of storing the link structures of the collection of web pages (Mukhopadhyay and Singh, 2004; Furnkranz, 1999). Maintaining a specific sequence based on ranking is a major task which is done by using SMACA. SMACA has its own attractor and transient states. All the states are uni-directional (refer to Figure 11). So, a state can have only one successor but multiple predecessors. Here, web pages are represented by transient states and related topics are shown as attractors. A mapping concept is introduced to refer the web pages as states of SMACA which is already discussed in Mukhopadhyay et al. (2006) and Kundu et al. (2007b). Non-reachable states are marked as higher ranked web pages and its successors are gradually lower in ranking.

For the formation of SMACA, the synthesis scheme is used as referred in Kundu et al. (2008a). Algorithm 9 shows the classifier design technique using SMACA concept.

Figure 11 Classifier design using SMACA



Algorithm 9 Classifier design technique based on page ranking

Input: A set of web pages of a particular domain containing specific topic	
Output: A classifier	
Step 1	Rank all web pages as per relevance using Algorithm 8 (Mukhopadhyay and Singh, 2004)
Step 2	Loop (Start)
Step 3	Place each web page in a sequence depending on their ranking
Step 4	Mark each web page as a distinct state
Step 5	Loop (Stop)
Step 6	Synthesise SMACA taking the assigned states as input (Kundu et al., 2007b, 2008a)
Step 7	Stop

Synthesis of SMACA demands development of a RV with group and non-group rules in some specific sequences. Identification method of such a sequence is described in Kundu et al. (2007b). A scheme is outlined in Kundu et al. (2008a) to recognise the sequence of rules in the RV that makes the CA a SMACA. The RV of an n-cell CA is denoted as $< R_0, R_1, \cdots, R_{n-1}>$, where ith cell is configured with R_i. In general, a non-linear 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 11 shows an arbitrary example of non-linear SMACA with its irregular structure as a classifier. States 2, 3, 4, 6 and 7 are attractors. State 0 is transient state. All other states are non-reachable states. Downloaded web pages are parsed using Parsing Agent to extract keywords as tokens. Assign uniquely generated key values of web pages as the attractors of the SMACA. Similarly, assign key values of tokens as non-reachable state or transient state of the SMACA. Instead of forward and inverted index files two sets of SMACAs are generated as discussed in Kundu et al. (2007b, 2008a) based on Algorithm 9.

Using this type of classifier concept, we have basically tried to modify the existing technology to store indexing data in an alternate way to reduce the storage size drastically. The access to forward and inverted indexed files adds to the time overhead. In this proposed technique, a non-linear single cycle multiple attractor cellular automata (SMACA) based design has been used that stores the classified data in an implicit manner, minimising storage space with a minimum access time. Typically, multi-level indexing is used within search engines. By using SMACA, multi-level indexing can be avoided completely.

We have achieved better results with respect to crawling time and indexing storage space for the GF(2^P) ranking system-based classifier. The experimental results are shown in the next section.

5 Experimental results

This section shows a comparison graph of timing between single crawler, parallel crawler and hierarchical crawler. Single crawler takes the most time for downloading the web pages, since it downloads the web pages one after another. A single thread is used to crawl the web pages. In case of parallel crawler, predefined numbers of web pages are crawled and further downloaded in parallel fashion in ideal condition. But, in real time, the web pages are crawled concurrently having some time delay. Parallel processors have been utilised in this case to handle the situation. Time and money have been saved using parallel crawling architecture. Hierarchical crawling is actually the advanced form of parallel crawling. Here, everything is dynamic. Numbers of threads are selected in

runtime to facilitate search engine oriented crawling. Figure 12 shows the comparison of timing between different types of crawlers. From the diagram, it is clear that overall timing is least in case of hierarchical crawling, since it selects the specific resources based on the criteria. The numbers of threads are not fixed.

Figure 12 Comparison between single, parallel and hierarchical crawling

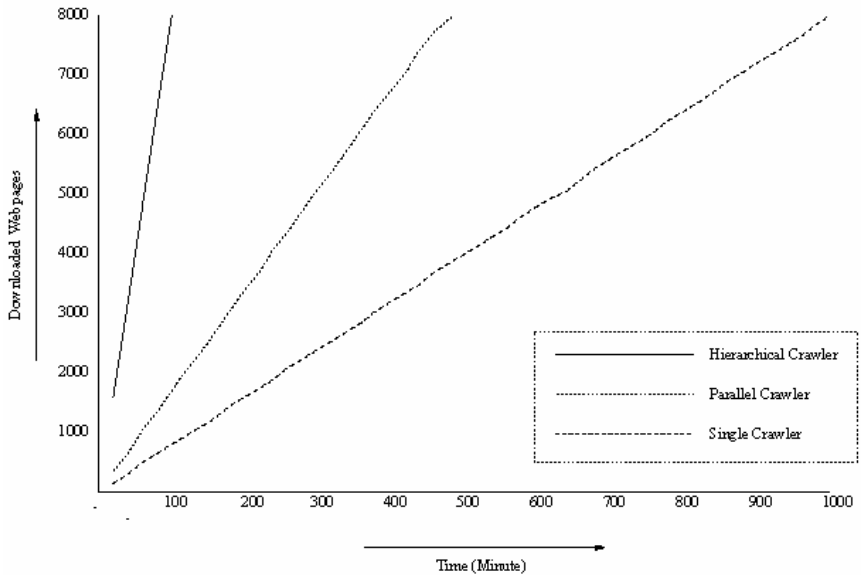
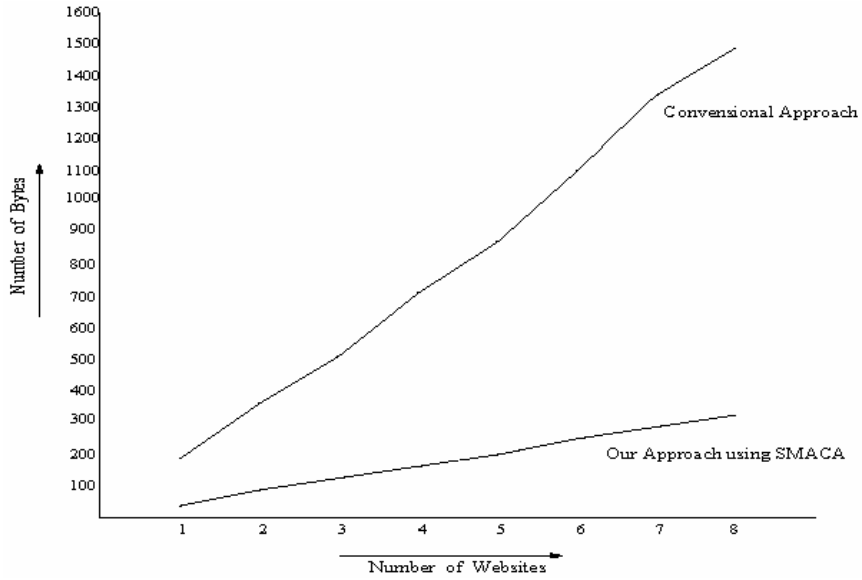
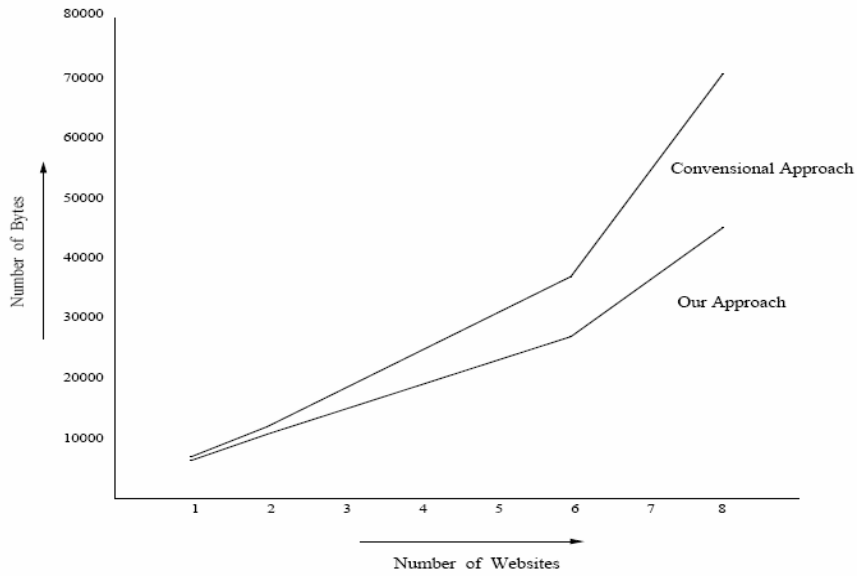


Figure 13 Comparison between typical indexing storage space requirement and our method using SMACA in forward indexing



This section also reports a brief study on non-linear cellular automata-based designing on storage of hypertext data while building a web search engine. Our experiment shows that it takes less storage space while searching through WWW as referred in Figure 13 and Figure 14.

Figure 14 Comparison between typical indexing storage space requirement and our method using SMACA in inverted indexing



6 Conclusions

In a conventional search engine, page ranking of web pages are used for better search result. Our current approach is to categorise the web pages into very broad categories along with page ranking at the time of classifier formation. The same approach could also be used to classify the web pages into more specific categories by changing the criteria. While searching through search engine, it will be much easier to find the desired result with less time complexity by exploiting our approach. Using cellular automata in web page classification is an entirely new concept that has been successfully implemented in this work which requires less storage space. Overall, we have succeeded to contribute in several parts of search engine following cellular automata as required depending on the strategy.

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.

Attardi, G., Gulli, A. and Sebastiani, F. (1999) 'Automatic web page categorization by link and context analysis', in *European Symposium on Telematics, Hypermedia and Artificial Intelligence*, Varese.

Available at http://www.yahoo.com.

Boldi, P., Codenotti, B., Santini, M. and Vigna, S. (2002) 'Ubicrawler: a scalable fully distributed web crawler', in *Proc. AusWeb02, The Eighth Australian World Wide Web Conference*.

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', in *IEEE Computer*, August, Vol. 32, No. 8, pp.60–67.

Chaudhuri, P.P., Chowdhury, D.R., Nandi, S. and Chatterjee, S. (1997) *Additive Cellular Automata, Theory and Applications, Vol. 1*, in IEEE Computer Society Press, Los Alamitos, California, Vol. ISBN-0-8186-7717-1.

Davison, B.D. (2000) 'Topical locality in the web', in *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000)*, ACM, July, pp.272–279, Athens, Greece.

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

Furnkranz, J. (1999) 'Exploiting structure information for text classification on the WWW', in *Intelligent Data Analysis*.

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

Kobayashi, M. and Takeda, K. (2000) 'Information retrieval on the web', in *ACM Computing Surveys*, ACM Press, Vol. 32, No. 2, pp.144–173.

Kundu, A., Dutta, R. and Mukhopadhyay, D. (2006a) 'An alternate way to rank hyperlinked web pages', in *9th International Conference on Information Technology*, 18–21 December, Bhubaneswar, India, IEEE Computer Society Press, New York, USA.

Kundu, A., Dutta, R., Mukhopadhyay, D. and Kim, Y. (2006b) 'A hierarchical web page crawler for crawling the internet faster', in *IEEE/CBNU/IEEK International Conference on Electronics & Information Technology Convergence*, 8 December, Republic of Korea.

Kundu, A., Dutta, R. and Mukhopadhyay, D. (2007a) 'Converging cellular automata techniques with web search methods to offer a new way to rank hyper-linked web-pages', in *International Symposium on Information Technology Convergence*, 23–24 November, IEEE CPS Publishing Services, Jeonju, Republic of Korea.

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

Kundu, A., Dutta, R. and Mukhopadhyay, D. (2008a) 'Design of SMACA: synthesis & its analysis through rule vector graph for web based application', in *International Journal of Intelligent Information and Database Systems*, Vol. 2, No. 4, Inderscience Publication, Europe.

Kundu, A., Pal, A.R., Sarkar, T., Banerjee, M., Mandal, S., Dattagupta, R. and Mukhopadhyay, D. (2008b) 'An alternate downloading methodology of web-pages', in *7th Mexican International Conference on Artificial Intelligence*, 27–31 October, IEEE CS, Mexico.

Kundu, A., Dutta, R., Dattagupta, R. and Mukhopadhyay, D. (2009) 'Mining the web with hierarchical crawlers – a resource sharing based crawling approach', in *International Journal of Intelligent Information and Database Systems*, Vol. 3, No. 1, Inderscience Publication, Europe.

Kwok, J.T. (1998) 'Automated text categorization using support vector machine', in *Proceedings of ICONIP'98, 5th International Conference on Neural Information Processing*.

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

Mukhopadhyay, D. and Biswas, P. (2005) 'FlexiRank: an algorithm offering flexibility and accuracy for ranking the web pages', in *Proceedings of the International Conference on Distributed Computing and Internet Technology, Lecture Notes in Computer Science Series*, 22–24 December, Springer-Verlag, India..

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

Mukhopadhyay, D., Giri, D. and Singh, S.R. (2003) 'An approach to confidence based page ranking for user oriented web search', in *ACM SIGMOD Record*, June, Vol. 32, No. 2.

Mukhopadhyay, D., Kundu, A., Dutta, R. and Kim, Y. (2006) 'An idea to minimize memory requirement and redundancy adopting cellular automata while building index file by web search engine', in *The 6th International Workshop MSPT 2006 Proceedings*, 20 November, Youngil Publication, Republic of Korea, ISBN 89-8801-90-0, ISSN 1975-5635.

Pinkerton, B. (1994) 'Finding what people want: experiences with the web crawler', in *Proceedings of the First World Wide Web Conference*, Geneva, Switzerland.

Risvik, K.M. and Michelsen, R. (2002) 'Search engines and web dynamics', in *Computer Networks*, June, Vol. 39.

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

Wong, W. and Fu, A.W. (2000) *Incremental Document Clustering for Web Page Classification*, in Chinese University of Hong Kong, July.