Mining the web with hierarchical crawlers – a resource sharing based crawling approach

Anirban Kundu* and Ruma Dutta

Netaji Subhash Engineering College,
West Bengal University of Technology,
West Bengal-700 152, India
and
Web Intelligence & Distributed Computing Research Lab,
(WIDiCoReL), Green Tower C-9/1, Golf Green,
Calcutta-700095, India
E-mail: anik76in@gmail.com E-mail: rumadutta2006@gmail.com
*Corresponding author

Rana Dattagupta

Jadavpur University,
West Bengal-700 032, India
E-mail: rdattagupta@cse.jdvu.ac.in

Debajyoti Mukhopadhyay

Calcutta Business School,
Diamond Harbour Road, Bishnupur,
West Bengal-743 503, India
and
Web Intelligence & Distributed Computing Research Lab
(WIDiCoReL), Green Tower C-9/1, Golf Green,
Calcutta-700095, India
E-mail: debajyoti.mukhopadhyay@gmail.com

Abstract: An important component of any web search engine is its crawler, which is also known as robot or spider. An efficient set of crawlers make any search engine more powerful, apart from its other measures of performance, such as its ranking algorithm, storage mechanism, indexing techniques, etc. In this paper, we have proposed an extended technique for crawling over the World Wide Web (WWW) on behalf of a search engine. This is an approach with multiple crawlers working in parallel combined with the mechanism of focused crawling (Chakrabarti et al., 1999a, 2002; Mukhopadhyay et al., 2006). In this approach, the total structure of any website is divided into several number of levels based on the hyperlink-structure for downloading web pages from that website (Chakrabarti et al., 1999b; Mukhopadhyay and Singh, 2004). The number of crawlers of each level is not fixed, rather dynamic in this context. It is determined at execution time on demand basis using threaded program based on the number of hyperlinks of a specific web page. This paper also proposes a focused hierarchical crawling technique, where crawlers are created dynamically at runtime for different domains to crawl the web pages with the essence of resource sharing.

Keywords: seed queue; SQ; single crawler; SCrw; parallel crawler; PCrw; hierarchical crawler; HCrw; focused hierarchical crawler; FHCrw; domain specific focused hierarchical crawler; DFHCrw.

Reference to this paper should be made as follows: Kundu, A., Dutta, R., Dattagupta, R. and Mukhopadhyay, D. (2009) 'Mining the web with hierarchical crawlers – a resource sharing based crawling approach', *Int. J. Intelligent Information and Database Systems*, Vol. 3, No. 1, pp.90–106.

Biographical notes: Mr. Anirban Kundu is a Senior Lecturer in Information Technology department of Netaji Subhash Engineering College, Calcutta, West Bengal, India. He passed BE Mechanical from Islamiah Institute of Technology, Bangalore University in the year 1999. He did Post Graduate Diploma in Financial Management from Management Studies Promotion Institute (registered under the Societies Registration Act of 1860) in the year 2001. He completed his MTech (IT) from Bengal Engineering and Science University, Shibpur in the year 2003.

Mrs. Ruma Dutta is doing her research work at the Web Intelligence & Distributed Computing Research Lab (WIDiCoReL) under Dr. Debajyoti Mukhopadhyay and Dr. Rana Dattagupta. She is also working as an Assistant Professor in the Computer Science and Engineering (CSE) Department of Netaji Subhash Engineering College, Calcutta, West Bengal, India.

Dr. Rana Dattagupta is working as a Professor in the Computer Science and Engineering (CSE) Department of Jadavpur University, Calcutta, West Bengal, India.

Dr. Debajyoti Mukhopadhyay is a Professor and Head of Information Technology and Management Information Systems at Calcutta Business School. He was Founder Director of Web Intelligence & Distributed Computing Research Lab (WIDiCoReL). He was a Full Professor of Computer Science & Engineering at West Bengal University of Technology affiliated Engineering colleges during 2001–2008. He was a Visiting Professor in Division of Electronics & Information Engineering at Chonbuk National University in Republic of Korea (2006–2007). He had taught at Stevens Institute of Technology, New Jersey (1982–1984) and at Bengal Engineering College, West Bengal (1980–1981).

1 Introduction

Crawling is the preliminary stage creating and maintaining a search engine for downloading web pages from the internet. These crawlers are mainly used by a search engine for searching web pages in order to download the target web pages from the internet space. After downloading, the web pages are further parsed and tokenised for building the database at server-side (Brin and Page, 1998; Glover et al., 2002).

Web crawler for any search engine searches for web pages. It also performs optimisation through indexing, tokenising and finding keywords for the search engine. It searches for index page for getting the valuable information about the particular website. Index page means the home page of a website. A search engine basically monitors the

information field of the internet utilising crawlers (Flake et al., 2000; Furnkranz, 1999) by verifying in-links and out-links of web pages.

Crawling of hyperlinks would take place till all appropriate web pages that comprise a website are crawled and indexed. Only those web pages that contain either a direct or indirect hyperlink would be crawled and/or indexed. Any link that a user establishes should transfer the user directly to the concerned website to enable viewing. This type of viewing without the imposition of any frames, browser windows or third-party content is referred to as 'direct hyperlink'. 'Indirect hyperlink' targets have a hyperlink reference in their link blocks.

Web pages that are 'orphaned' have no inbound links directed to them and would not be indexed at all. In other words, orphan web pages have no links. The basic idea of crawler is to crawl the web directly or indirectly through other hyperlinks. Crawler collects all the important information about web pages. Crawler keeps a copy of the visited web pages (Kundu et al., 2006; Mukhopadhyay and Biswas, 2005; Mukhopadhyay et al., 2003a, 2003b). A shorter version of this work was proposed in Kundu et al. (2006). In the proposed technique, crawlers are utilised to accumulate web pages of different domains under separate databases (Arasu et al., 2001; Davison, 2000). Existing crawling techniques typically follow either single crawling or parallel crawling concepts available in the market. In this paper, we are going to propose a hierarchical method for crawling web pages from the web. Hierarchical method means a technique with a hierarchical view. Here, a number of crawlers have been utilised based on the requirement of the level of hierarchy for this purpose. The 'number of crawlers' is calculated using the directory structure of the concerned web directory. It is a scalable crawling scheme which minimises network utilisation along with time realisation. Websites are considered as tree structured documents and web pages are considered as different nodes of the particular tree (website). The rest of the paper is organised in this way: in Section 2, the background work is shown; the proposed approach is described in Section 3; in Section 4, the experimental results are shown in brief; and the conclusion of this work is shown in Section 5.

2 Background

In this section, two existing crawler based techniques have been shown for searching distinct web pages using hyperlinks. Crawler typically downloads web pages to store into the repository. Our aim is to show the superiority of hierarchical crawler (HCrw) with respect to single crawler (SCrw) and parallel crawler (PCrw). So, SCrw and PCrw have been implemented as the basis of our enhanced approach which is shown in the next section.

In rest of the paper, certain alpha-numeric symbols have been used in Figure 1 to Figure 5.

a1 search for web page

a2 download web page

S1 crawler requests for a seed URL

S2 seed queue (SQ) grants a URL for the crawler

M1 Module 1 or crawling module

M2 Module 2 or parsing module

M3 Module 3 or URL transfer module

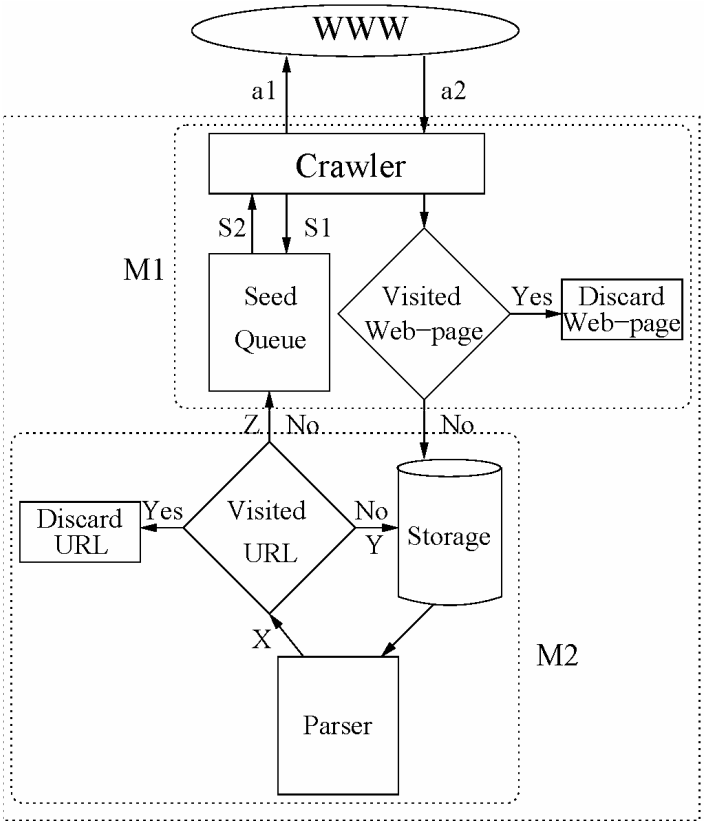
X extracted URLs

Y saving new URLs to database

Z transmit new URLs to SQ.

M1 and M2 both are dependent on each other. M1 cannot perform without the seed URL(s) which comes from M2 as a result of parsing. Similarly, M2 cannot parse the web page until it is downloaded and saved by the crawler into storage. So, M2 is also dependent on M1. These two modules work in tandem.

Figure 1 Existing single crawling system



Algorithm 1 describes the functionality of the SCrw. It checks whether a crawled web page has been already visited or not by searching the existing database. Visited web pages are discarded, whereas new web pages are being saved into the repository. These downloaded web pages are then parsed to extract the URLs. One Parser and one SQ are required in SCrw. After parsing, the algorithm validates the already extracted

URLs as mentioned in Figure 1. Here, validation means the checking if the extracted URLs are already in the existing database. Visited URLs are discarded and new URLs are stored into the storage as well as within SQ. Fast lexical analyser generator (FLEX) has been used as a Parser tool for the experiment.

Definition 1 SQ – a queue, containing number of URLs as seed for downloading web pages from the internet, is known as SQ.

Definition 2 Valid topic – valid topic means related topic of the specific web page; whether the contents are related to the topic of interest.

Definition 3 SCrw – a SCrw is a program module for downloading web pages one at a time.

Algorithm 1 Single crawling system

Input A set of seed URLs within SQ

Output Storage of downloaded web pages within the server

Step 1 Crawler starts crawling with seed URLs

Step 2 Crawler downloads the web page taking URL from the SQ

Step 3 Check whether the URL of the 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 SQ as well as in storage

Step 9 If (condition for continuing the downloading process satisfies)

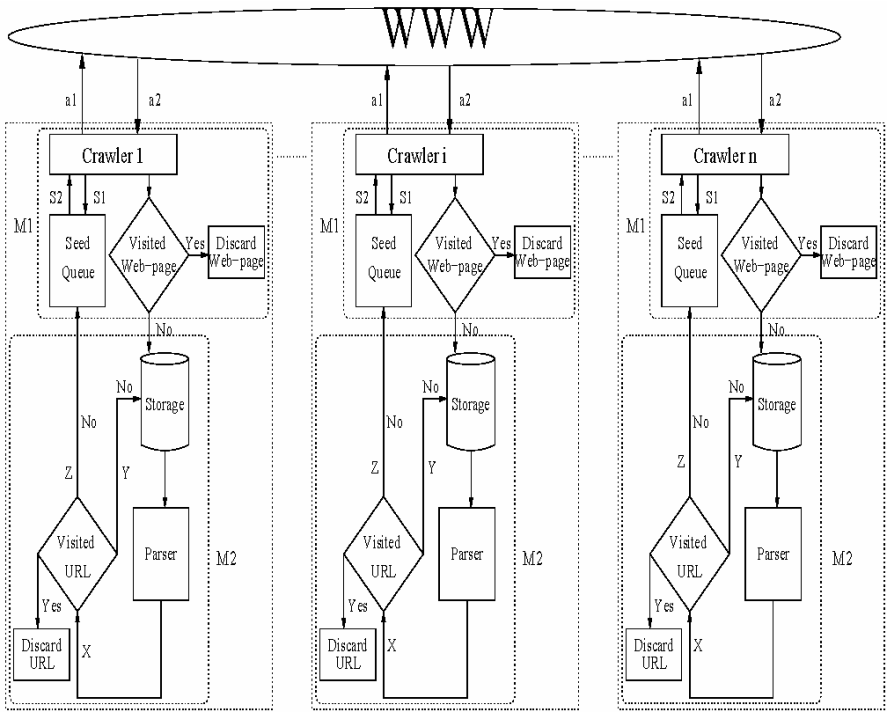
Step 10 Then go to Step 2

Step 11 Stop.

Parallel crawling system is shown in Figure 2 in a same fashion and the functionality is described in Algorithm 2. Unlike SCrw, PCrw consists of several numbers of crawlers working in parallel to achieve the goal. The number of crawlers is fixed or predetermined depending on the software vendor. The 'N' number of Parser and SQ are required in PCrw. Thus, a number of web pages can be downloaded at a time using PCrw which is not a dynamic system in true sense. For example, a PCrw system can download and further process the web pages 'N' times faster than SCrw using the 'N' (N> = 1) number of crawlers in the ideal situation. So, the system overhead of individual machine is much lesser in case of PCrw, though the number of crawlers cannot be changed at runtime. In this type of PCrw system, the total workload is divided into several segments in a distributed manner. At any time instance, there may be some idle crawlers depending on the requirement. Consider a situation where out of the 'N' number of crawlers, only 'n' number of crawlers is executing for downloading web pages from WWW. Then, the

'N – n' number of crawlers is wasting the system resources. This wastage is later removed from the system using HCrw concept.

Figure 2 Existing parallel crawling system



Definition 4 PCrw – a set of crawlers working concurrently is known as PCrw.

Algorithm 2 Parallel crawling system

Input A set of seed URLs within different SQs

Output Storage of downloaded web pages within the servers

Step 1 A set of crawlers start crawling with seed URLs

Step 2 Each crawler downloads the web page taking URL from their respective SQ

Step 3 Check whether the URL of the downloaded web page is already visited or not

Step 4 If visited, discard the web page and go to Step 10

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

Step 9 Then save the hyperlinks within the respective SQ and storage

Step 10 If (condition for continuing the downloading process satisfies)

Step 11 Then go to Step 2

Step 12 Stop.

3 Proposed approach

In our approach, the crawling system is further enhanced to achieve better performance with respect to the time only. This paper is not concerned about the resources since it is well available in the present situation. We are calling it HCrw. Here, a set of crawlers are generated dynamically at runtime depending on the number of seed URLs in SQ. Each crawler downloads a specific web page within its life time and then kills itself as described in Algorithm 3. A counter is used to calculate the total number of URLs in SQ at any time instance.

Definition 5 Life cycle of dynamic crawler – life cycle of crawler refers to the time period between the crawler creation and its destruction.

Algorithm 3 Life cycle of dynamic crawler

Input A set of seed URLs

Output Downloaded web pages

Step 1 Check the number of URLs within SQ

Step 2 Generate the same number of crawlers in runtime

Step 3 Assign each seed (URL) to a specific crawler

Step 4 Download all web pages

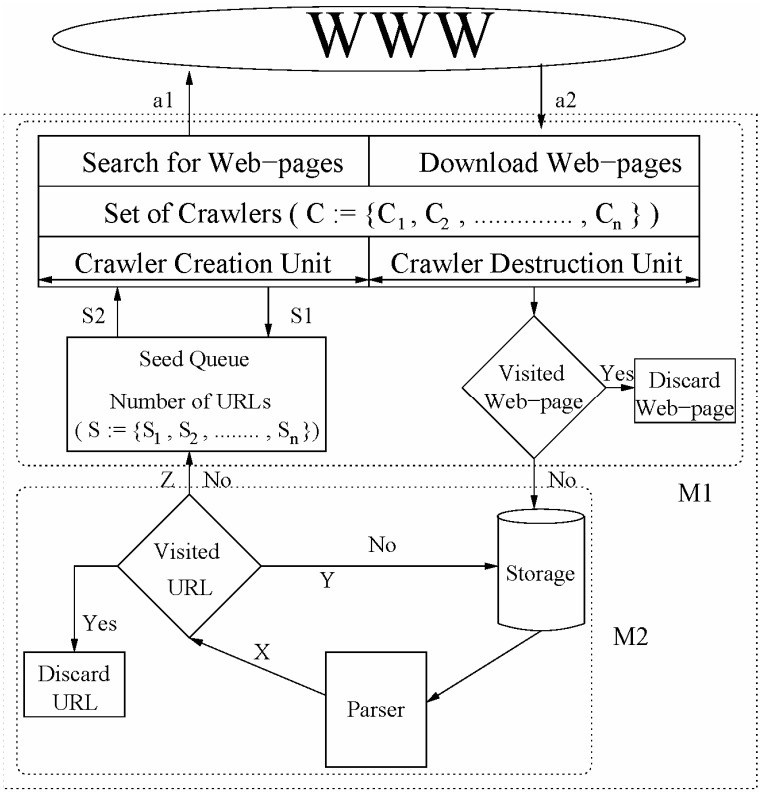
Step 5 Kill all crawlers

Step 6 Stop.

HCrw is described in Algorithm 4. The creation of dynamic crawlers plays an important role in this algorithm for downloading web pages based on their depth within the concerned web directory of a particular website. Parser module waits for new web pages for extracting each and every link/URL from it. Only one Parser and one SQ are required for HCrw. After parsing, these extracted URLs are being compared with the existing URL database of the repository. Those URLs, which are mismatched with the existing database, are being saved into the URL database as well as into the SQ for downloading web pages at the next level using a set of dynamic crawlers. HCrw is the combination of dynamic PCrw and depth level of searching. Here, depth level ensures the penetration limit. For example, if depth = 0, then only the initial URLs of SQ are downloaded through the 'N' number of dynamic crawlers. If depth = 1, then the downloaded web pages of depth = 0 are parsed by Parser module and the new distinct URLs are saved within SQ for downloading the next set of web pages through dynamically created crawlers as discussed in Algorithm 4 (Figure 3). Like this, at any depth, the HCrw system generates the 'N' number of crawlers, does the needful work and then kills the crawlers.

Downloading operation is done parallel; but, the number of crawlers are not fixed rather it depends on the number of links available within SQ at runtime. So, there is no chance of wastage of the system resources. Idle crawler in such a situation is impossible since crawlers are created in a dynamic sense.

Figure 3 Proposed hierarchical crawling system



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

Definition 6 HCrw – HCrw is a set of dynamically created crawlers whose number of instance is dependent on the number of seed URLs. One crawler can download only one web page using its assigned URL from the SQ in its life time. After downloading the web pages, these crawlers are destroyed automatically.

Definition 7 Depth – level of searching through WWW for downloading the required web pages in a hierarchical fashion is referred to as depth of searching.

Algorithm 4 Hierarchical crawling system

Input A set of seed URLs within SQ 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 Call Algorithm 3

Step 4 Check whether the URLs of the downloaded web pages are already visited or not

Step 5 Discard already visited web pages

Step 6 Save new web pages

Step 7 Hyperlinks of all saved web pages are extracted using Parser tool

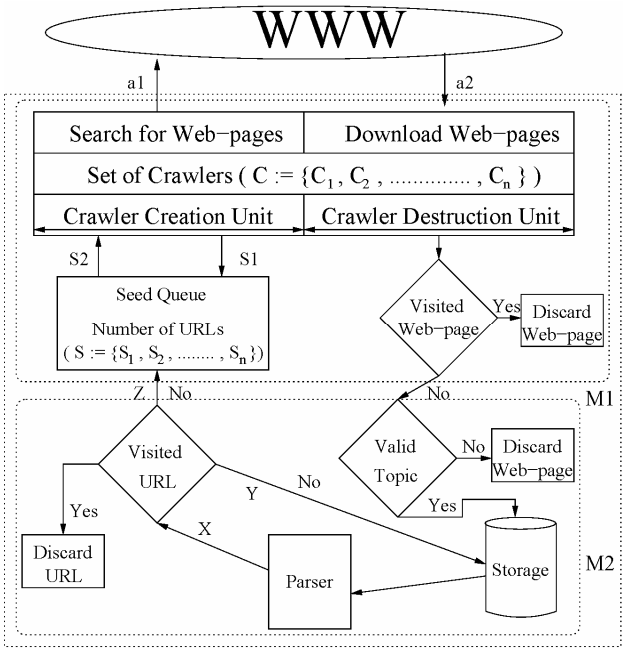
Step 8 Check whether the extracted URLs are already visited or not

Step 9 Save those extracted hyperlinks (URLs), which are still not visited, within SQ as well as in storage

Step 10 Increment i by 1

Step 11 Stop.

Figure 4 Proposed focused hierarchical crawling system



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

Focused crawling is introduced in HCrw for topic based searching. Focused hierarchical crawler (FHCrw) is introduced in this scenario to penetrate for better results. The aim of our FHCrw is to selectively search for web pages that are relevant to a pre-defined set of topics. The FHCrw 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. Since a crawler may download web pages which are not required for a specific work of interest, FHCrw is structured using Algorithm 5 for better performance. In this algorithm, an additional control point is introduced. This control is useful to check whether the downloaded web page has got the valid topic or not as shown in Figure 4. Here, valid topic means the specific area of interest. The design is modified for building FHCrw offering higher performance. The number of Parser module and SQ required in FHCrw are only one. Many times, a URL may have several synonymous addresses. Our proposed approach has taken care of the situation to download and save distinct web pages. If the downloaded web page has another URL which is already visited through a crawler, then the downloaded web page would be discarded. 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'.

Definition 8 FHCrw – an HCrw being used for a specific topic is known as FHCrw.

Algorithm 5 Focused hierarchical crawling system

Input A set of seed URLs within DQ 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 Call Algorithm 3

Step 4 Check whether the URLs of the downloaded web pages are already visited or not

Step 5 Discard the visited web pages

Step 6 New web pages are being checked for valid topic

Step 7 Discard the web pages which do not contain a valid topic

Step 8 Save remaining web pages

Step 9 Hyperlinks of all saved web pages are extracted using Parser tool

Step 10 Check whether the extracted URLs are already visited or not

Step 11 Save those extracted hyperlinks, which are still not visited, within SQ as well as in storage

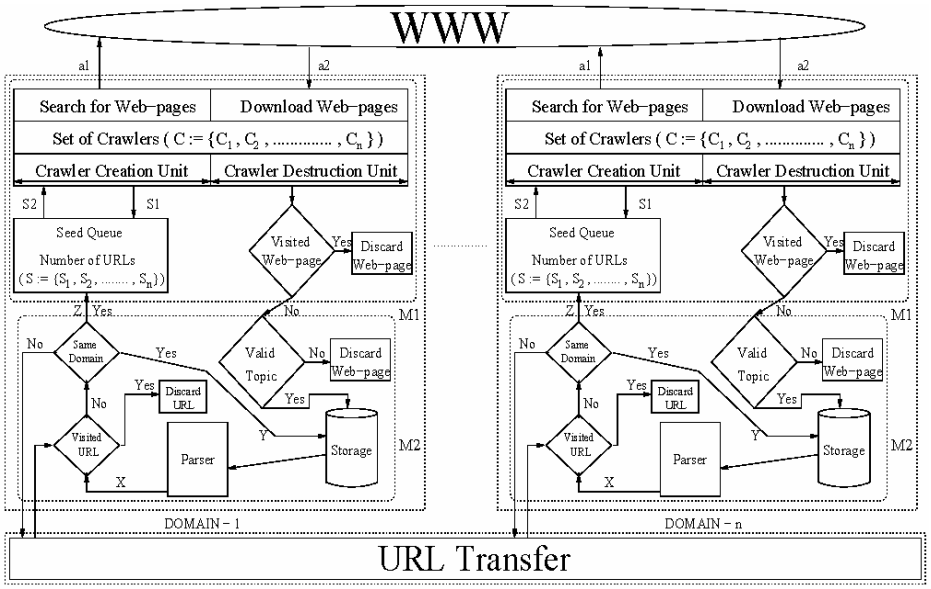
Step 12 Increment i by 1

Step 13 Stop.

Finally, different computer machines are assigned for storing web pages of different domains using Algorithm 7. Thus, the whole network load is dispersed into distinct domain loads. By HCrw technique, web coverage is more within a specific time compared to SCrw or PCrw. Since we have used distinct crawler modules to download

and store web pages of various domains, there is no chance of overlapping (Figure 5). A 'URL Transfer Program' module is used to transfer data from one domain to another using Algorithm 6. After parsing the downloaded web pages, there can be many URLs containing the address of other domain. That is why this type of transfer module is a necessity while designing the whole system. So, dynamic assignment is done for allocating URLs within the SQ of distinct domains. In the domain specific focused hierarchical crawler (DFHCrw), there are one SQ and one Parser module for each domain related system.

Figure 5 Proposed domain specific focused hierarchical crawling system



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

Algorithm 6 URL transfer

Input Parsed URL and the corresponding downloaded web page of a particular domain

Output Transfer of URL to domain based SQ

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 SQ of related domain

Step 6 Stop.

Algorithm 7 Domain specific focused hierarchical crawling system

Input	A set of seed URLs of different domains within different SQs and depth of searching
Output	Storage of focused downloaded web pages within servers of respective domains
Step 1	Initialise i with 0
Step 2	Continue loop until i > Depth
Step 3	Call Algorithm 3
Step 4	Check whether the URLs of downloaded web pages are already visited or not
Step 5	Discard the visited web pages
Step 6	New web pages are being checked for valid topic
Step 7	Discard web pages which do not contain a valid topic
Step 8	Save remaining web pages
Step 9	Hyperlinks of all saved web pages are extracted using Parser tool
Step 10	Check whether the extracted URLs are already visited or not
Step 11	If not visited, then check for the domain
Step 12	If extracted hyperlinks belong to same domain
Step 13	Then save the hyperlinks within the respective SQ and storage
Step 14	If extracted hyperlinks belong to different domains
Step 15	Then call Algorithm 6
Step 16	Increment i by 1
Step 17	Stop.

Based on the discussion so far, the role of the HCrw is summarised in the following terms. Search engine downloads the websites (WS_i) which are already registered with them through the registration procedure.

Theorem 3.1 Time requirement in HCrw for downloading a set of web pages is minimum compared to SCrw and PCrw.

Let

$$WS_i = Registered\ website\ \forall i,$$

where $1 \leq i \leq n$ and $n = Number\ of\ registered\ websites$.

In subsequent discussion, a level of hierarchy of a particular website is denoted by 'L' and the number of web pages at any L is n_L.

Consider, ${WP_L}^j$ is the jth web page at level 'L'. So, ${WP_L}^j \in WS_i \forall i,$ where $1 \leq L \leq n$ and $1 \leq j \leq n_L \forall L$.

Therefore, for downloading web pages at any level 'L':

Minimum number of crawlers required, $Crw_{min} = 1$

and

Maximum number of crawlers required, $Crw_{max} = Max(n_L) \forall L$,

Time taken by a crawler to download a web page $= t_{WP_L{}^j}$.

Therefore, the maximum time required by a set of crawlers to download all web pages at level L, $t_{max} = Max(t_{WP_L{}^j}) \forall j$.

The total time required to download all the web pages of a particular website through SCrw, PCrw and HCrw are shown respectively:

$$\text{Time required by } SCrw = \sum_{L=1}^{n} \sum_{j=1}^{n_L} t_{WP_L{}^j}, \tag{1}$$

$$\text{Time required by } PCrw = \sum_{L=1}^{n} \sum_{j=1}^{n_L/X} Max(t_{WP_L{}^{(j-1)*X+1}}, \ldots, t_{WP_L{}^{j*X}}), \tag{2}$$

where $X = fixed\ or\ predefined\ number\ of\ crawlers$;

$$\text{Time required by } HCrw = \sum_{L=1}^{n} Max(t_{WP_L{}^1}, \ldots, t_{WP_L{}^{nL}}). \tag{3}$$

By enlarging equation (3), we got the following modified equation:

$$\text{Time required by } HCrw = \sum_{L=1}^{n} \sum_{j=1}^{n_L/n_L} Max(t_{WP_L{}^{(j-1)*n_L+1}}, \ldots, t_{WP_L{}^{j*n_L}}). \tag{4}$$

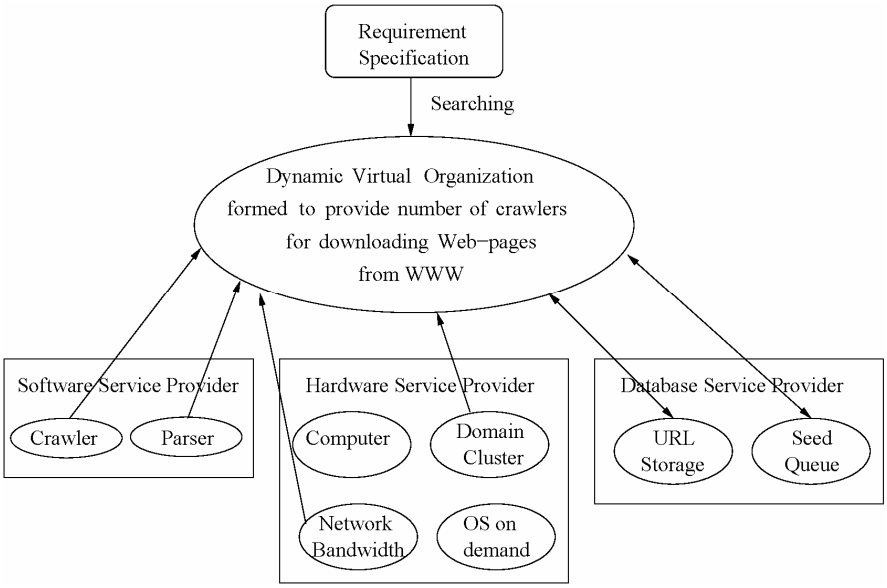
Since, $n_L > n_L / X > n_L / n_L$, where $1 < X < n_L$.

Thus, from equations (1), (2) and (4), we conclude that the time requirement of HCrw is lesser than SCrw and PCrw. Hence proof.

This theorem is valid under all conditions due to the fact that 'X' should be greater than '1' and lesser than 'n_L'. We are not considering the case of '$X = n_L$' because in practical scenario, it is not possible to predict the number of actual links within a web page which is not visited at the moment. So, only one thing can be done that for each link, system may generate a different crawler. For example, if there are '30' web pages of a particular online tutorial, SCrw downloads all the web pages one after another up to '30' times. On the other hand, PCrw downloads these '30' web pages within $30/X$ times. Here, 'X' is the number of predetermined crawlers set for this operation. Generally, 'X' ranges from 'two' to 'ten'. Now, if HCrw comes into picture, it downloads '30' web pages at a time if available the bandwidth permits. The most important point is that a user does not know the total number of web pages required for downloading the whole tutorial at the starting point. So, the user cannot specify the exact number of crawlers. In the case of HCrw, there is no need to specify the number of crawlers at the starting point of the downloading operation as crawlers are generated dynamically on demand basis for each level of hierarchy.

HCrw are well-managed through parallel and distributed system on demand basis. This system enables sharing, selection and aggregation of geographically distributed autonomous resources dynamically at runtime based on its availability, capability, performance, cost and users' quality-of-service requirements as shown in Figure 6. In our approach, the dynamic virtual organisation enables the sharing, selection and aggregation of a wide variety of resources including computers, storage systems, data sources and specialised devices which are geographically distributed. Software, hardware and database service providers have been used in this context.

Figure 6 Required infrastructure in resource sharing for controlling HCrw



4 Experimental results

Table 1 Sample downloaded websites

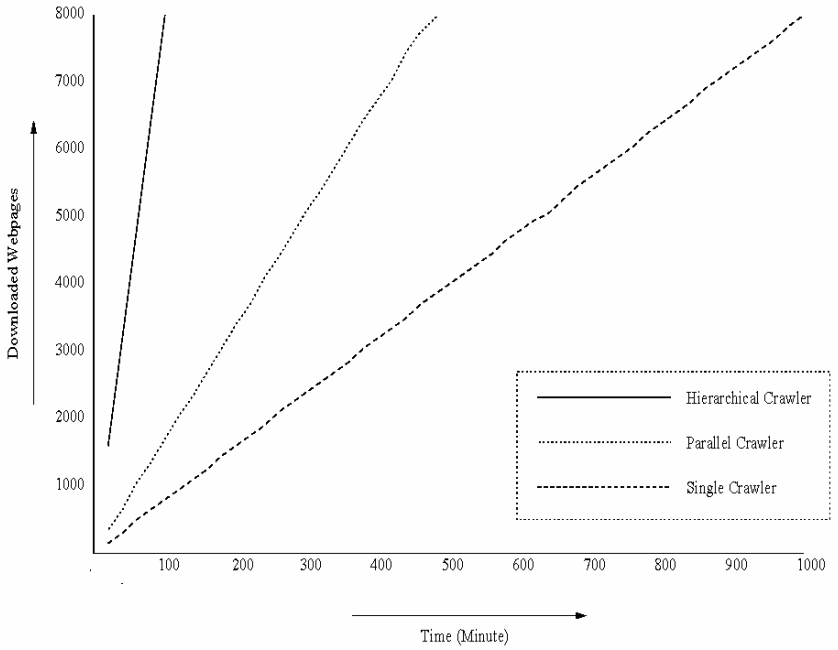
Sl. no.	*Website name*	*Total number of web pages downloaded*
1	freshersworld.com	4,994
2	theatrelinks.com	469
3	indiagsm.com	34
4	rediff.com	163
5	w3.org	2,333
6	indiafm.com	7,087
7	nokia-asia.com	193
8	amazon.com	349

Table 2 Comparative study on time taken by SCrw, PCrw and HCrw

Sl. no.	*SCrw*	*PCrw*				*HCrw*
		2	*4*	*6*	*8*	
1	9 hr. 30 min.	7 hr. 47 min.	6 hr. 5 min.	4 hr. 21 min.	2 hr. 40 min.	58 min.
2	44 min.	37 min.	30 min.	23 min.	17 min.	10 min.
3	3 min.	2 min. 30 sec.	2 min.	1 min. 30 sec.	1 min.	30 sec.
4	3 min.	2 min. 28 sec.	1 min. 56 sec.	1 min. 24 sec.	52 sec.	20 sec.
5	6 hr.	4 hr. 55 min.	3 hr. 51 min.	2 hr. 46 min.	1 hr. 42 min.	37 min.
6	7 hr.	5 hr. 47 min.	4 hr. 33 min.	3 hr. 20 min.	2 hr. 6 min.	53 min.
7	2 hr.	1 hr. 39 min.	1 hr. 19 min.	58 min.	37 min.	17 min.
8	2 hr. 58 min.	2 hr. 26 min.	1 hr. 54 min.	1 hr. 23 min.	45 min.	19 min.

Note: Number of PCrw = two to eight for this experimentation

Figure 7 Timing diagram of single, parallel and hierarchical crawling system



In this section, the experimental results are shown in Table 1 and Table 2. In Table 1, a few numbers of websites are enlisted along with the number of downloaded web pages up to Depth Level 4 of the concerned web hierarchy. These web pages are downloaded through SCrw, PCrw and HCrw separately for comparing the time taken by each type of crawler in Table 2. In Figure 7, it is clearly shown that the HCrw performs far better than SCrw and PCrw. Since the HCrw exploits maximum bandwidth at any time instance,

more number of web pages can be downloaded within a particular time with respect to the SCrw and PCrw. So, the web coverage is more in the case of HCrw. The following configurations of the computer systems have been used for the experimentation: processor – Pentium 4 (P4); processor speed – 2.8 GHz; primary memory (RAM) – 512 MB; hard disk – 80 GB; and internet connection speed – 512 kbps, 1:1 RF line.

5 Conclusions

In a search engine, typically SCrw or PCrw is used for downloading web pages. Due to the exponential growth of WWW every day, there exists a need for a faster mechanism to efficiently download web pages in a quickest possible time. The existing practices use crawlers in parallel in the best case scenario. We have shown in this paper a new concept of hierarchical design of crawlers targeting web pages to minimise time requirement while crawling through the internet. These crawlers are generated at runtime dynamically using the number of URLs present within the SQ by threaded program. These URLs can be at any depth level of concerned hierarchy of the hyperlinked structures. After downloading the specific web pages, the crawlers are destroyed automatically. At any time instance, the maximum number of web pages available for concurrent downloading depends on the allowable bandwidth of the system.

Acknowledgements

The authors would like to acknowledge the many helpful suggestions of the participants of *IEEE/CBNU/IEEK International Conference on Electronics and Information Technology Convergence* (EITC 2006), Jeonju, Republic of Korea 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., Berg, M. and Dom, B.E. (1999a) 'Focused crawling: a new approach to topic-specific web resource discovery', *Proceedings of the Eighth International World Wide Web Conference*, Elsevier, Toronto, Canada, pp.545–562.

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

Chakrabarti, S., Punera, K. and Subramanyam, M. (2002) 'Accelerated focused crawling through online relevance feedback', *Proceedings of the Eleventh International World Wide Web Conference*, 7–11 May, Honolulu, Hawaii, USA.

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

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.

Furnkranz, J. (1999) 'Exploiting structure information for text classification on the WWW', *Intelligent Data Analysis*, pp.487–498.

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

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

Kundu, A., Dutta, R., Mukhopadhyay, D. and Kim, Y-C. (2006) 'A hierarchical webpage crawler for crawling the internet faster', *International Conference on Electronics & Information Technology Convergence, EITC 2006 Proceedings*, 8 December, Yang Dong Publication, ISSN 1975-809X, Republic of Korea, pp.61–67.

Mukhopadhyay, D. and Biswas, P. (2005) 'FlexiRank: an algorithm offering flexibility and accuracy for ranking the web pages', *Proceedings of the International Conference on Distributed Computing and Internet Technology*, 22–24 December, Lecture Notes in Computer Science Series, Springer-Verlag, India, pp.308–313.

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

Mukhopadhyay, D., Giri, D. and Singh, S.R. (2003a) 'A confidence based methodology to deduce user oriented page ranking in searching the web', *Proceedings of the International Conference ITPC 2003*, 23–26 May, Nepal.

Mukhopadhyay, D., Giri, D. and Singh, S.R. (2003b) 'An approach to confidence based page ranking for user oriented web search', *ACM SIGMOD Record*, June, Vol. 32, No. 2, pp.28–33.

Mukhopadhyay, D., Mukherjee, S., Ghosh, S., Kar, S. and Kim, Y-C. (2006) 'Architecture of a scalable dynamic parallel web crawler with high speed downloadable capability for a web search engine', *The 6th International Workshop MSPT 2006 Proceedings*, November, Youngil Publication, Republic of Korea, pp.103–108.