A COMPARATIVE STUDY OF SCHEDULING ALGORITHMS FOR WEB CRAWLING USING VB.NET TECHNOLOGY

*Sushil Kumar, #Dr. Anuj Kumar

*Research Scholar, CMJ University, Shillong Meghalaya -793003, India

ABSTRACT

Under the present study, Web Crawler simulator has been designed than analyze the different web crawling algorithm to evaluate their performance. Web crawler is a computer program or software. Web crawler is an essential component of search engines, data mining and other Internet applications. Scheduling Web pages to be downloaded is an important aspect of crawling. Previous research on Web crawl focused on optimizing either crawl speed or quality of the Web pages downloaded. While both metrics are important, scheduling using one of them alone is insufficient and can bias or hurt overall crawl process. ***This paper is all about the comparative study of scheduling algorithm for Web Crawling using VB.NET Technology****.*

INTRODUCTION

A web-crawler is a program/software or automated script which browses the World Wide Web in a methodical, automated manner. The structure of the World Wide Web is a graphical structure; the links given in a page can be used to open other web pages. Actually Internet is a directed graph, webpage as node and hyperlink as edge, so the search operation could be abstracted as a process of traversing directed graph. By following the linked structure of the Web, we can traverse a number of new web-pages starting from a starting webpage. Web crawlers are the programs or software that uses the graphical structure of the Web to move from page to page. Such programs are also called wanderers, robots, spiders, and worms. Web crawlers are designed to retrieve Web pages and add them or their representations to local repository/databases. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages that will help in fast searches. Web search engines work by storing information about many web pages, which they retrieve from the WWW itself. These pages are retrieved by a Web crawler (sometimes also known as a spider), which is an automated Web browser that follows every link it sees. Web are programs that exploit the graph structure of the web to move from page to page. It may be observed that `crawlers' itself doesn’t indicate speed of these programs, as they can be considerably fast working programs.

Web crawlers are software systems that use the text and links on web pages to create search indexes of the pages, using the HTML links to follow or crawl the connections between pages.

SIMULATOR DESIGN

This section covers the high level design and detailed design of the Web Crawler. Next section presents the high level design of the Web Crawler in which summarised algorithmic view of proposed crawler is presented. And after high level section, next section describes the General architecture of the Web Crawler Simulator, technology and programming language used, user interface of simulator or crawler and performance metric concepts. The proposed crawler simulator imitates the behaviour of various crawling scheduling algorithms. This section briefly describes the overall working of simulator in an algorithmic notation. Algorithm describes below presents the high level design of Web Crawler Simulator

Step 1. First of all accept the URL and use this URL as the Seed or Acquire URL of processed web document from processing queue.

Step 2. Add it to the Frontier.

Step 3. Now pick the URL from the Frontier for Crawling.

Step 4. Use this URL and Fetch the web page corresponding to that URL and store this web document.

Step 5. Parse this web document's content and extract set of URL links.

Step 6. Add all the newly found URLs into the Frontier.

Step 7. Go to **step 2** and repeat while the Frontier is not empty.

Step 8. Output desired statistics. Step 9**. Exit.**

Thus a crawler will recursively keep on adding newer URLs to the database repository of the search engine. So we can see that the main function of a crawler is to add new links into the frontier and to select a new URL from the frontier for further processing after each recursive step.

GENERAL ARCHITECTURE OF THE CRAWLING SIMULATOR

Below figure shows the flow of the crawler simulation architecture [Ard¨o A]. The simulator is designed so that all the logic about any specific scheduling algorithm is encapsulated in a different module that can be easily plugged into the system

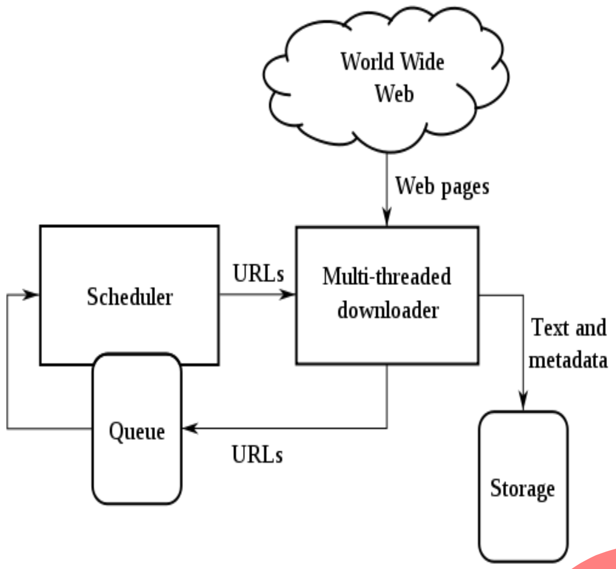


Figure 1

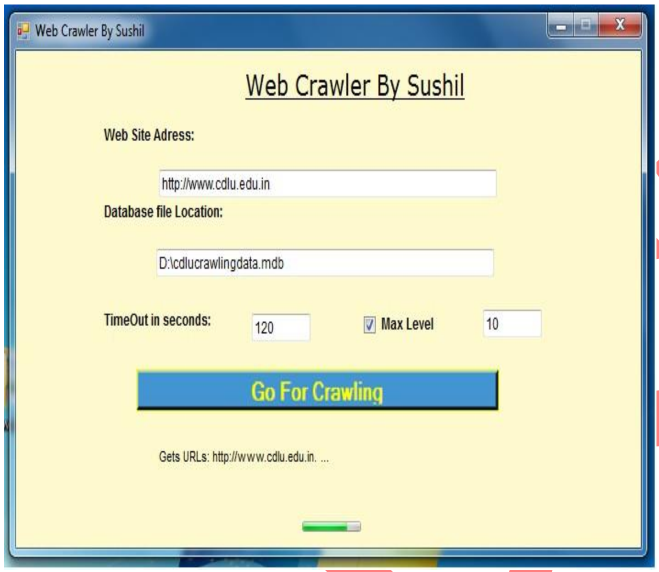
All crawling modules share the data structures needed for the interaction with the simulator. The simulation tool maintains a list of unvisited URLs called the frontier. This is initialized with the seed URLs specified at the configuration file. Besides the frontier, the simulator contains a queue. It is filled by the scheduling algorithm with the first k URLs of the frontier, where k is the size of the queue mentioned above, once the scheduling algorithm has been applied to the frontier. Each crawling loop involves picking the next URL from the queue, fetching the page corresponding to the URL from the local database that simulates the Web and determining whether the page is relevant or not. If the page is not in the database, the simulation tool can fetch this page from the real Web and store it into the local repository. If the page is relevant, the outgoing links of this page are extracted and added to the frontier, as long as they are not already in it. The crawling process stops once a certain end condition is fulfilled, usually when a certain number of pages have been crawled or when the simulator is ready to crawl another page and the frontier is empty. If the queue is empty, the scheduling algorithm is applied and fills the queue with the first k URLs of the frontier, as long as the frontier contains k URLs. If the frontier doesn't contain k URLs, the queue is filled with all the URLs of the frontier.

CRAWLER USER INTERFACE

The foremost criterion for the evaluation of crawling algorithm is the number of relevant pages visited that are produced by each crawling algorithm under same set of Seed URLs. Than simulator has been designed to study the behaviour pattern of different crawling algorithms for the same set of starting URLs.

Page rank, relevant pages visited and the order in which a set of pages is downloaded are considered to evaluate the performance of crawling policy and crawler. During the

implementation process, we have taken some assumptions in to account just for simplifying algorithms implementation and results.
Figure shown below is the snapshot of the main user interface of the Web Crawler simulator, which is designed in the VB.NET using ASP.NET Window Application project type, for crawling a website or any web application using this crawler internet connection must required and as input use URL in a format like: **http://www.google.com** or **http://google.com** and set location and name of database for saving crawling results data in MS-Access database.



Web Crawler By Sushil

Web Site Adress:

http://www.cdlu.edu.in

Database file Location:

D:\cdlucrawlingdata.mdb

TimeOut in seconds: 120 ☑ Max Level 10

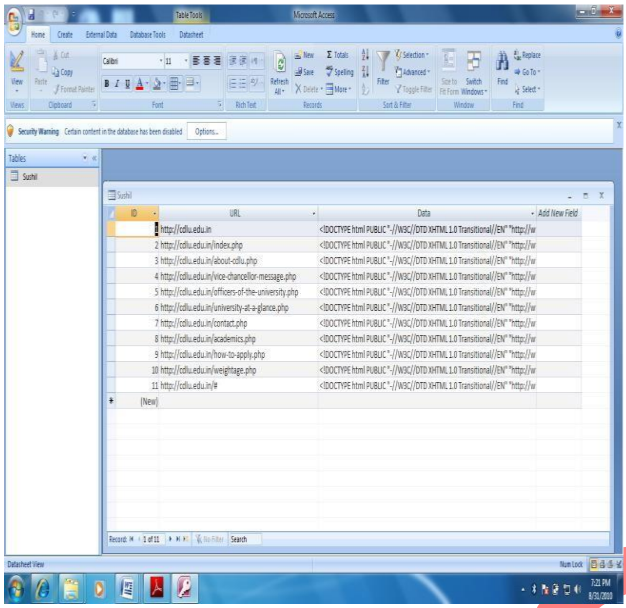
Gets URLs: http://www.cdlu.edu.in. ...

Figure 2

At each simulation step, the scheduler chooses the topmost Website from the queue of Web sites and sends this site's information to a module that will simulate downloading pages from the Website. For this Simulator uses the different crawling scheduling policies and save data collected or downloaded in MS Access Database in a table with some data fields which are ID, URL and Data.

CRAWLING RESULT

The best way to compare the result of different policies is to present them in form of table depicting the result in the form of Rows and columns. Output of
first simulated algorithm, Breadth First algorithm is shown below as a snapshot.



Sushil

ID	URL	Data	Add New Field
1	http://cdlu.edu.in	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
2	http://cdlu.edu.in/index.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
3	http://cdlu.edu.in/about-cdlu.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
4	http://cdlu.edu.in/vice-chancellor-message.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
5	http://cdlu.edu.in/officers-of-the-university.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
6	http://cdlu.edu.in/university-at-a-glance.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
7	http://cdlu.edu.in/contact.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
8	http://cdlu.edu.in/academics.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
9	http://cdlu.edu.in/how-to-apply.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
10	http://cdlu.edu.in/weightage.php	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
11	http://cdlu.edu.in/#	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://w	
(New)			

Figure 3

The simulator uses the breadth first algorithm and crawled the website for the URL http://www.cdlu.edu.in . The working of any Breadth-First algorithm is very simple. It simply works of first come first serve. Crawling start with URL http://www.cdlu.edu.in. After processing this URL, its child link inserted into the Frontier. Again the next page is fetched from the Frontier and is processed, its children inserted into Frontier and so on. This procedure continues until the Frontier gets empty.

Breadth-First is used here as a baseline crawler; since it does not use any knowledge about the topic, and its performance is considered to provide a lower bound for any of the more sophisticated algorithms.

Second optimal algorithm is Best First; in this the preference of next page to be approached depends upon the relevancy of that page. Best First traversing from the same URL as Breadth First algorithm for http://www.cdlu.edu.in. Its relevancy is set highest (2 in this case). Now the relevancy of seed children comes out to be 0.1 and 1.0 respectively. Along with respective relevancies, seed children are inserted in frontier. Every time, the page with highest relevancy value is picked from the Frontier. The parent relevance decides which page will be selected next. The page with the highest parent relevance value is selected from the Frontier every time. Third crawling algorithm is Breadth First with time constraints and crawled result using this algorithm for the same seed URL. It is analysed from the produced results that each policy behaves differently with same seed URL.

RESULT FORMAT

Downloaded data page by page after crawling from the website http://www.cdlu.edu.in and store in the database table's Data Column field using the format shown below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Language" content="en-us" />

<meta http-equiv="X-UA-Compatible" content="IE=7" />

<meta http-equiv="Content-Type" content="text/html; charset=windows-1252" />

<meta name="verify-v1" content="uNh2/LFAVip3xI8N/LIVD63/1YquyPWEyzegOUv80Ls="
/>

<title>Welcome to CDLU, Sirsa</title>

<meta name="keywords" content="Chaudhary Devi Lal University, CDLU, University in Sirsa,
Distance education, Sirsa university, university in haryana, devi Lal university, CDLU Sirsa, Sirsa
university, Tau devi lal, choudhary Devi lal, university of haryana, MCA in sirsa, Mass
communication in sirsa, M.A. in Sirsa, M.Com in sirsa" />

…..

…..

…..

</body>

</html>
```

This format shows the crawled data for the home page of the website.

Graphical Representation

The simulation results on taking 100 pages visited are summarized in Table 1. Figure 4 shows that the Breadth-First with time constraints retrieved 37.691 relevant Web pages (37,691% of all pages visited) compared with 35.090 (35,090%) by PageRank, 33.930 (33,930%) by Best First and 28.797 (28,797%) by Breadth- First. Figure 5 shows the evolution of the precision rate for each one of the scheduling algorithms. At the beginning, PageRank was the one that achieved the best performance, but at the end the result was rather similar to Breadth First with time constraints. Breadth First with time constraints had a more constant performance and at the end its result was the best. Best First achieved the worst performance during almost all the execution, but at the end it reached a precision rate rather close to Breadth First with time constraints and

PageRank. Breadth First had a regular performance, but it was not as good as Breadth First with time constraints and PageRank.

	No of Pages Visited	No of Relevant Pages Visited	Precision
Breadth First	100.000	28.797	28,797%
Best First	100.000	33.930	33,930%
PageRank	100.000	35.090	35,090%
Breadth First with Time Constraints	100.000	37.691	37,691%

Table 1: Crawling Results

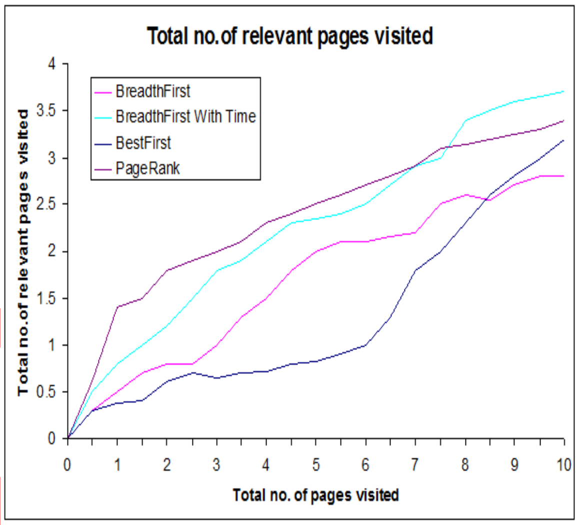


Figure 4: Total number of relevant pages visited

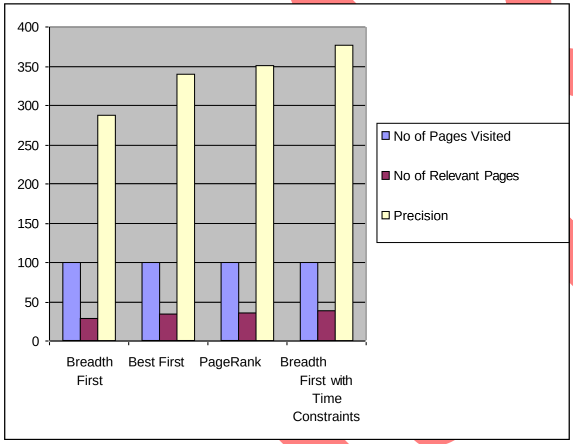


Figure 5: Percentage of relevant pages among pages visited

CONCLUSION

Internet is one of the easiest sources available in present days for searching and accessing any sort of data from the entire world. The structure of the World Wide Web is a graphical structure, and the links given in a page can be used to open other web pages. In this dissertation, Graphical structure is used to process certain traversing algorithms used in the search engines by the Crawlers. Each webpage can be considered as node and hyperlink as edge, so the search operation could be abstracted as a process of traversing directed graph. By following the linked structure of the Web, crawler can traverse a number of new web pages starting from a Starting webpage. Web crawlers are the programs or software that uses the graphical structure of the Web to move from page to page. In this dissertation, firstly discussed about Internet, Search Engines, Crawlers and then Crawling Algorithms in brief.

There are number of crawling strategies used by various search engines. The basic crawling approach uses simple Breadth First method of graph traversing. But there are certain disadvantages of BFS since it is a blind traversing approach. To make traversing more relevant and fast, some heuristic approaches like best first, are followed. The results of all the crawling approaches are giving different results.

After analysing the results and findings of the crawler it might be concluded that, **the crawler developed could be really helpful and useful to study the performance of different crawling algorithms in terms of precision, without taking into consideration the time. But it is not a difficult task to compute the time that a crawling algorithm takes in this simulation framework. According to the results, it can be observed that the number of relevant pages retrieved by each one of the crawling algorithm doesn't depend on the set of starting URLs rather than used crawling algorithm**. But this affirmation may be confirmed running more experiments with different values for each one of the input parameters. The weakness of the crawler lies in the time to compute the algorithms. New algorithms need to be adjusted to the shared data structures and parameters. In addition, each of the proposed scheduling algorithms has its own special features. For this reason, it is difficult to implement each of them in the most efficient way.

The future work will be directed in two directions. On one hand, and the most obvious, the implementation of more scheduling techniques in order to find one with a really good performance. Probably, this algorithm will assign *a* specific score to each individual outgoing link*;* instead of inherit it from the page itself. When this algorithm will be found, it will have to be tested in a real crawler in order to verify its performance. On the other hand, the attempt to find a measure that given the precision, the number of fetched pages and the time for scheduling specifies which is the best combination of these three parameters, in order to know which is the best performance.

REFERENCES

1. http://en.wikipedia.org/wiki/Web_crawler#Examples_of_Web_crawlers

2. http://www.chato.cl/papers/castillo04_scheduling_algorithms_web_crawling.pdf

3. http://ieeexplore.ieee.org/iel5/2/34424/01642621.pdf

4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.9569&rep=rep1&type=pdf.

5. http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf

6. Marc Najork, Allan Heydon SRC Research Report 173, "High-Performance Web Crawling", published by COMPAQ systems research center on September 26, 2001.

7. Sergey Brin and Lawrence Page, "Theanatomy of a large-scale hyper textual Web search engine", In Proceedings of the Seventh International World Wide Web Conference, pages 107–117, April 1998.

8. [Ard¨o A]. (2005). "Combine Web crawler," Software package for general and focused Web-crawling. http://combine.it.lth.se/.