Survey on Smart Web Crawler Algorithm Based on Semantic Search Engine

Muneer Bani Yassein, Ismail Hmeidi, Yaser Khamayseh, Wail Mardini, Heba Al-Jarrah
Jordan University of Science and Technology,
Irbid, Jordan
{Masadeh, hmeidi, yaser, mardini}@just.edu.jo,
hebaatta96@gmail.com

Dragana Krstic
Faculty of Electronic Engineering, University of Niš,
Niš, Serbia
email:dragana.krstic@elfak.ni.ac.rs

***Abstract*—With the increasing amount of information and inflation of pages in the World Wide Web, it becomes difficult for the user to retrieve relevant information it needs. In other words, it leads to a big challenge for a search engine, on how to extract the exact results that match with the user query. Hence, we need smart programs such as Web Crawlers to overcome this problem. Web crawlers use different kinds of algorithms, some of these algorithms using the heuristic function to achieve the goal in an efficient manner such as A*, Adaptive A*, Best-search algorithm, and some other algorithms are not using a heuristic function such as Depth-first search and Breadth-first search. This survey covers these two types of algorithms and focuses on the mechanisms and the difference between them.**

Keywords-Web Crawling Algorithms; Uniform Resource Locator (URL); Best-first search; Heuristic function; Web Crawler.

I. INTRODUCTION

Web crawlers, also known as web spiders and web robots, are programs that visit websites and read and download their pages and other information to create entries for a search engine index. The crawler crawls with a list of Uniform Resource Locators (URLs) to visit, called the seeds. As the crawler visits these URLs, it identifies all the URL hyperlinks in the page and adds them to the list of the visited URLs (queue), called the crawl frontier. In its simplest form, the crawler starts from the seed page and then uses external links inside it to access other pages. The process is repeated with new pages that offer more external links to follow until a sufficient number of pages is identified, or a higher-level goal is reached Web crawlers are an essential component of web search engines, where they are used to gather the corpus of web pages indexed by the search engine. Moreover, they are used in many other applications that process large numbers of web pages, such as web data mining, and comparison-shopping engines. [1]. The crawler can start from any seed URL, but this is not enough to reach all the web pages. The pages referenced by the seed URLs should not reference it back to them, otherwise it will end up in an infinite loop, and this is a crucial factor to be considered. Hence, it is essential to start from a proper seed URL. For example, Bing or Google search engines can be used to get seed URL by merely entering the keywords into them and consider their resulting links as seed URLs. It is not possible to cover and index all the websites in the World Wide Web for a particular search entry. A web crawler always downloads a fraction of the web pages that contain the most relevant pages and not just a random sample of the Web [2] [3]. The relevance of a page depends on some factors such as the number of visits on the page. Web crawlers employ different strategies for selecting the websites to be downloaded, such as depth-first, breadth-first, best first, A* and other algorithms that will be covered in this survey.

The crawler aims to crawl the World Wide Web and report back some essential data and then perform an initial data analysis using some additional data before it stores the collected data. To achieve this purpose, the crawler that crawls the Internet must have the following basic features to serve their purpose [5]:

a) Robustness: The Web contains loops which are aims to mislead the crawler to recursively crawl a particular domain and get stuck in one single domain.

b) Distributed: The crawler should be able to work in a distributed manner to crawl the Internet as quickly as possible.

c) Scalable: The crawler should have the flexibility to add new machines whenever necessary.

d) Extensible: The crawlers should be able to adapt to the increasing amount of data formats on the Web.

e) Quality: The crawler should be able to distinguish between useful and useless information. In other words, they should be able to filter out the relevant content.

f) Freshness: The crawler should make sure that the concepts on the search engine are the latest and relevant to the present context.

This paper is organized like this: in Section 2, the architecture of web crawler is presented, and different types of web crawlers are listed. The web crawler strategies are specified in Section 3, and conclusion and future work are shown in the last, fourth section.

II. Architecture and Types of Web Crawlers

The architecture in Fig. 1 [6] provides an outline of the main modules of a web crawler. URL frontier contains a list of visited URLs (queue) to be fetched. A DNS resolution module converts the URLs to its equivalent IP addresses. Fetch module that uses the HTTP request to retrieve web pages associated with its URL. A parsing module extracts the query text and set of links from a fetched web page. Finally, a duplicate elimination module checks to determine whether an extracted link is already in the URL frontier or has recently been fetched.

There are different types of crawler; in this survey, some of them will be covered.

A. *Focused Crawler*

This kind of crawlers works by gathering documents that have specific areas of interest. In other words, it collects the documents similar to each other. The main thing to be kept in mind is that the page is downloaded after it is estimated that the page is similar and relevant to the given topic. The main advantage of this type is the availability of the URLs without downloading the page to predicate the similarity of not traversal page, meaning fewer costs.

B. *Distributed Crawler*

This type of crawlers distributes the work to several other crawlers to perform the task at the same time. Then, it collects all downloaded pages from all crawlers and sends them to a central indexer where links are extracted and sent via the URL server to the crawlers. The main advantage of distributed crawler is that it is strong. It can withstand system crashes and other types of problems and can adapt to different crawling requirements.

C. *Parallel Crawler Algorithm*

Parallel Crawler (PC), which means more than one crawler, runs multiple processes in parallel, where PC depends on freshness and Selection Page [7]. The main idea for PC is to maximize the download rate and achieve minimum overhead by using parallelization. We can find more than one crawling processes for the same URL. PC can be distributed in different places or at the same local network [8]. In [7] and [8], the authors presented different algorithms that will help to achieve high performance.

III. Web Crawler STRATEGIES

In this section, several search algorithms used by Web crawlers will be enumerated and described.

A. *Breadth First Search*

It is the simplest form of crawling algorithms, which starts with the root URL links and searches on the connected links URL (in the same level). This algorithm is also known as Blind search algorithm. That is because it is not considering any information about the topic and means the breadth-first search requires the maximum amount of searching time.

On each page the crawler visits, all external links (hyperlinks) will be extracted from the web pages, and all these links will be distributed in the searching tree on the leaves of the current node [9].

In Fig. 2, we show an example where the search starts from the root URL (i.e., Seed page 1) and then collects URL page 1.1 and page 1.2 which get searched in sequence at the same level, then pages 1.1.1, 1.1.2, 1.2.1 and 1.2.2 are searched in order [9].

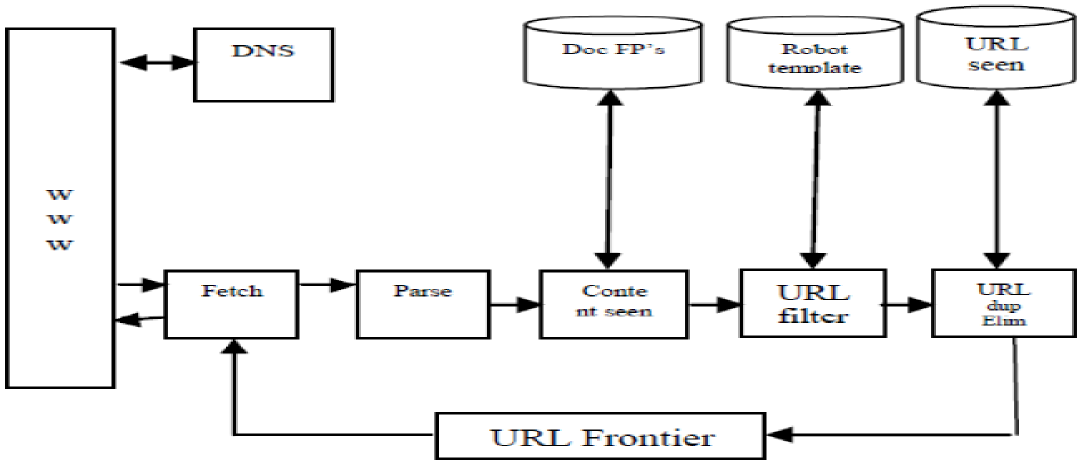

Figure 1. The architecture of crawler.

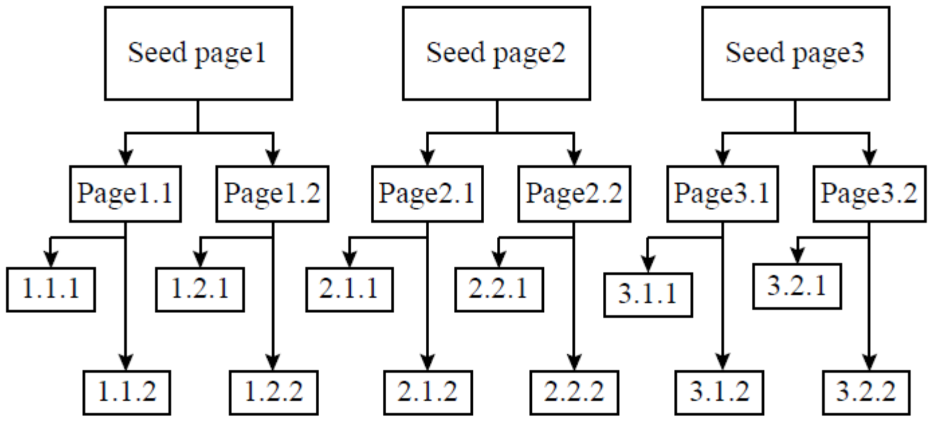

Figure 2. Web crawler forest.

B. *Depth First Search*

In this search algorithm, the crawler starts from the root URL and traverses to child URLs in the deep form. If there is more than one child, the priority will be going to the most left child and then keep performing deep traversal until it receives no more nodes for searching steps. After that, backtrack to the previous node to check whether there exists another unvisited node, and then continue similarly. This algorithm might end up in an infinite loop in case of large number of branches.

In the example in Fig. 2, the search starts with root URL (i.e., Seed page 1), and then the search proceeds by downloading pages 1.1, and 1.1.1, then page 1.2 and its own child's are searched in order.

C. *Best First Search*

Best First Search is an informed search algorithm based on estimate function which predicts the similarity between the candidate URLs to pick the best one to fetch. This estimated value could be the score or rank of relevancy. There are many algorithms to apply the classifier for these heuristic values, such as the cosine similarity, Support Vector Machine (SVM) and string matching used for scoring [10].

Unlike Uninformed search, the best first search algorithm uses heuristic function to improve its result as much as possible by exploring the node with the best score first [5].

D. *A* Search*

A* algorithm uses the best first search algorithm to estimates the lowest total cost of any solution path. It combines the features of uniform-cost search and greedy (heuristic) search to compute the optimal path solution. The cost associated with a node is $f(n)= g(n) + h(n)$, where $g(n)$ is the cost of the path from the initial state to the goal node, and $h(n)$ is the heuristic estimate for the cost of the path from node n to the goal node. $f(n)$ can be referred to as the relevancy cost of each link.

The A* algorithm gives the best result of relevant retrieve information with the lowest time compared with other algorithms that don't use a heuristic function.

E. *Adaptive A* Search*

Adaptive A* Search is an informed heuristic search. In each iteration, it improves the relevancy value of the Web page and uses this new update in the next iteration. At the beginning of the process, it takes more time because it stores the history of previous iterations with every search. Once it gathers sufficient knowledge about the relevant pages, it needs the same time as A* search, Best-first search and any other heuristic-based search algorithms.

F. *Page Rank Search*

Page Rank algorithm works by counting the number and quality of links to a page in order to determine how important the website is. So, as it is shown by the equation below, the Page Rank does not rank websites but is determined for each page individually

$$PR(P_1) = PR(T_1)/L(T_1) + ... + PR(T_i)/L(T_i).$$

To find the Page Rank for a page, called $PR(P_1)$, we need to find Page Rank of all pages T_i with inbound links and outbound links to page P_1. For example, page T_1, which has a link from P_1, is calculated. Then page $L(T_1)$ will give the number of outbound links to the page P_1. We do the same for T_2, T_3, up to page T_i, then the Sum of all the weighted Page Ranks of all pages is calculated.

G. *Path-Ascending Crawling Algorithm (PAC)*

The crawler tries to obtain more information for a specific website by crawling every path for that URL [11]. For example, in Fig. 3, we have a URL: http://llama.org/hamster/monkey/page.html, PAC will try to crawl /hamster/monkey/, /hamster/, and /. The added values for PCA algorithm become very efficient while finding the resource.

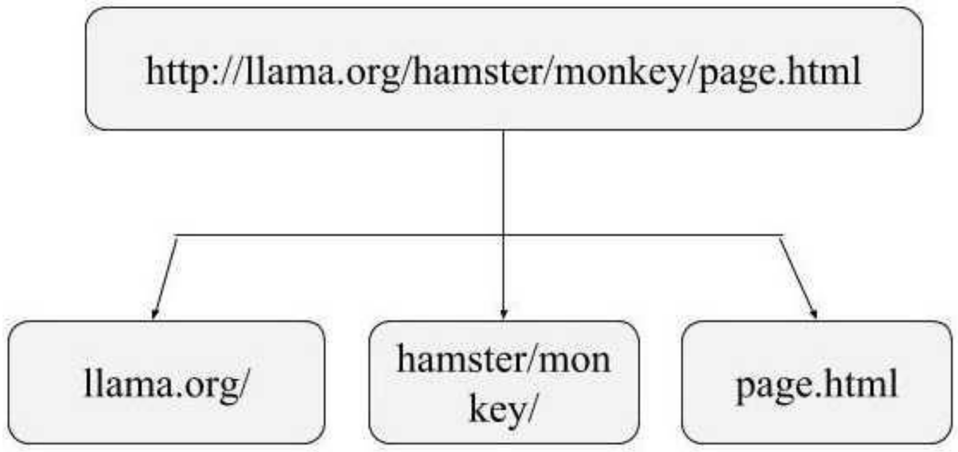

Figure 3. Path-Ascending Crawling Algorithm (PAC)

TABLE I. COMPARISON BETWEEN ALGORITHM

Algorithm indicator	Number of the relevant pages	Time to retrieve
Breadth-first search	Extremely small	Extremely large
Depth-first search	Quite small	Large
Best-first search	Large	Low
A* algorithm	Extremely large	Extremely Low
Adaptive A* search	Large	Low
Page Rank search	Large	Large

IV. CONCLUSION AND FUTURE WORK

Reducing the search time with the best relevant result is one of the main issues being targeted by search engines these days. This survey focuses on the Web crawling algorithms that use heuristic function and do not to retrieve relevant data. Breadth-first search is a blind algorithm and is not an efficient method. The depth-first search may end up in an infinite loop when the branches are too large. The Best first search and A* algorithm show nearly the same result with respect to search time. They can be improved by enhancing the heuristic function. Adaptive A* performs more efficiently when the users search at the same type of content because it depends on the history of searches, and with each search, the efficiency of the search will increase. This may consume some amount of time compared with the A* algorithm. All algorithms discussed in this paper are effective for search engine, however, the advantages favor algorithms that use the heuristic functions to retrieve the best relevant URLs and least response time.

REFERENCES

[1] M. Najork, Web Crawler Architecture, Microsoft Research, Mountain View, CA, USA

[2] V. Kancherla, A Smart Web Crawler for a Concept Based Semantic Search Engine, San Jose State University, 2014.

[3] D. Yadav, A. Sharma, and J. P. Gupta, “Parallel crawler architecture and Web page change detection”, WSEAS Transactions on Computers, issue 7, volume 7, July 2008, pp. 929-940.

[4] S. M. Pavalam, S. V. Kasmir Raja, M. Jawahar, and F. K. Akorli, “Web crawler in mobile systems”, International Journal of Machine Learning and Computing, International Journal of Machine Learning and Computing, vol. 2, no. 4, August 2012. pp. 531-534.

[5] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

[6] R. Thite, B. Pawar, T. Mode, and M. Mete “Survey paper on smart Web crawler”, International Journal of Novel Research in Computer Science and Software Engineering, vol. 3, issue 1, pp. 274-277.

[7] S. Chakrabarti, M. Van den Berg, and B. Dom, “Focused crawling: a new approach to topic-specific Web resource discovery”, Computer Networks, 31.11-16, 1999, pp. 1623-1640.

[8] N. Tyagi and D. Gupta. “A novel architecture for domain specific parallel crawler”, Department of Computer Engineering, Shobhit University, Meerut, India, 2010.

[9] Y. Yu, S. Huang, N. Tashi, H. Zhang, F. Lei, and L.Wu, “A survey about algorithms utilized by focused Web crawler”, Journal of Electronic Science and Technology, vol. 16, no. 2, June 2018.

[10] R. Dechter and Judea Pearl. “Generalized best-first search strategies and the optimality of A*”, Journal of the ACM, 32(3), pp. 505–536, 1985, doi:10.1145/3828.3830

[11] V. Cothey, “Web-crawling reliability”, Journal of the American Society for Information Science and Technology 55.14, 2004, pp. 1228-1238.

[12] R. Janbandhu, P. Prashant, and M. M. Raghuwanshi, “Analysis of web crawling algorithms”, International Journal on Recent and Innovation Trends in Computing and Communication, 2.3, pp. 488-492, 2014.