IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-04, Issue-09 E-ISSN: 2347-2693

Heuristic Approach for Designing a Focused Web Crawler using Cuckoo Search

Joy Dewanjee

Department of Computer Science and Engineering,
Guru Nanak Institute of Technology, India

Available online at: www.ijcseonline.org

Received: 26/Aug/2016 Revised: 08/Sept/2016 Accepted: 21/Sept/2016 Published: 30/Sep/2016

Abstract— In order to find a geographical location in the Globe, we usually follow the geographical map. By a similar analogy, a Web-page from the World Wide Web (WWW), we usually use a Web search engine. Web crawler design is an important job to collect Web search engine resources from WWW. Millions of searches are done every minute around the Globe. A better Web search engine resource leads to achieve a better performance of the Web search engine. WWW is a huge resource of information. However this information is often spread throughout the internet via many Web servers and hosts. Every day people are publishing their Web pages in the Internet, as a result the traffic overhead increases exponentially. In order to produce a more accurate result, I have been motivated to follow a heuristic approach to design a Web crawler, which produces the best optimized search result in minimal time. This paper has built an approach to generate the best result in by Cuckoo Searching so that time will be least. I have divided my approach in two parts. First part is implementation of the crawler, which includes "what to search for", "from where to search" and even filters the unwanted data. Second part proposed a string matching algorithm for producing the search result.

Keywords— Cuckoo search; DNS; meta-heuristic; optimization; pattern recognition; web crawling;

I. Introduction

The basic concept of web crawler is based upon the idea of fetching information from the World Wide Web by analyzing the different web pages and other resources. As being a focused crawler, it is provided with an initial clue which helps in the direction of search. Every individual search engine is categorized and modularized in a finite and sorted fashion on which the search engine mostly relies as it helps to provide the best possible optimized results. Crawlers are miniaturized block of codes that surf the web rather than the search engine, similarly to how a regular person would seek and proceed to links in order to view different pages. The crawlers are fed with a set of starting seed i.e. the Unified Resource Locator (URL), whose webpages need to be revived from the Web. The crawler retrieves the URLs or hyperlinks present in the retrieved pages, and gives this data to the crawler control module. Then this module decides which links to follow, and provides it to the crawlers. The crawler is made to work fast by using an optimized search technique.

This crawling technology used the cuckoo search algorithm [1] which is a recently formulated meta-heuristic optimization algorithm, suitable for explaining optimization problems. In order to increase the accurateness and convergence rate of this crawler, an augmented crawler algorithm is proposed which is based on the cuckoo search optimization technique. When the crawler [2] is using URL to look for any specific content, the basic motive of the cuckoo search algorithm is to minimize the number of URLs the crawler has to go through by securing a selected generation of the URLs which is claimed to provide a better result based on previous search statistics.

This focus based web crawler approach extracts files that contain similar keyword from the input fed by the user; which is implemented using breadth-first search. Now, pattern recognition is applied when the web pages are extracted. A file is given as input for the application of the pattern recognition algorithm. Here, pattern mostly resembles text and used to check the quantity of text available on the web page. The algorithm used for pattern search is Knutt-Morris-Pratt algorithm.

II. EXISTING WORK

In this section I have discussed some existing works carried out by the Web researchers.

A. *Web Crawling*

Web crawler plays fundamental role in making the Web easier to use for millions of people. A Web crawler is a program which crawls through the WWW on behalf of the search engine and downloads Web-pages in Web-page repository for further processing by the search engine. For a

Corresponding Author: *Joy Dewanjee, joydewanjee@ gmail.com*
Department of Computer Science and Engineering,
Guru Nanak Institute of Technology, India

Web crawler [3,4,5], Domain Name System (DNS) thread of an URL first checks with the DNS database in order to resolve the host name; if following is achieved, then the thread extracts the IP from the database; or else, DNS threads tends to get the IP from a DNS server. Then, a thread which is already read receives the resolved IP address and tries to establish an HTTP socket connection in order to seek for the web page. After fetching the web page, the web crawler scans the entire page content in order to look for any previously visited link. If nothing is found, then it retrieves and normalizes the URLs, checks the further possibility of crawling, and also determines whether they are previously visited or not. In order to reduce the server load some timestamp has been fed to the crawler, so that it crawls until the timestamp expires. If it is unable to fetch the result within the provided timestamp, it will give default message stating string not found; or else it will fetch the present links and display and store it in the output file.

Various types of Web-page crawling mechanism have already been introduced by the Web researchers. Significant resources of the network are used by the web crawlers to construct a comprehensive text index of the entire Web and thus keeping it up-to-date. Cho and his team [6] had estimated that the crawlers of big search engines can crawl up to ten million pages every day. To reduce the pressure on the network, Rajender and Satinder [7] proposed a novel mobile crawler system based on filtering of non-modified pages. Shkapenyuk and Suel [8], Paolo and co-workers [9] have proposed distributed Web crawlers. Edwards and his team [10] have proposed an adaptive model in order to optimize the efficiency of an incremental Web crawler. Marc and co-workers [11] have introduced a breadth-first crawling mechanism for yielding high quality Web-pages. Around 20 years ago, Pinkerton [12] has proposed a Web crawling mechanism for finding what people want. Later on lots of researches were carried out on focused crawling. Chakrabarti and his team [13], Altingövde and Ulusoy [14], Zong and co-workers [15], S Mewada [16], Pant and Srinivasan [17], Almpanidis and co-workers [18], were proposed some focus crawling mechanisms based on topic-specific Web resource discovery. Diligenti and co-workers [19], Bergmark and co-workers [20] have presented some efficient focus crawlering mechanism, which used in digital libraries. There are plenty of research work done by Web researchers on Web Crawler. I have also proposed a new approach of designing a focused Web crawler using Cuckoo Search.

B. *Brief Idea of Cuckoo search Algorithm*

The idea of most meta-heuristics [21] comes from the fact that they are able to imitate the best and most unique features in nature, especially in case of biological systems evolving from natural selection over the time span. The most important characteristics are selection of the fittest and adaptation to the environment which can be translated into two crucial characteristics of the modern meta-heuristics: intensification and diversification [21, 22]. Intensification is the idea of searching around the current best solutions and choose the best solutions; whereas diversification determines whether the algorithm can explore the search space more accurately and efficiently. Here I have considered the following cuckoo search algorithm [1] which is based on the behavior of cuckoo birds:

```
Objective function: f(x), x= (x¬1, x2,…, xd),
    • Generate an initial population of n host nests;
    • While (t<MaxGeneration) or (stop criterion)
          a. Get a cuckoo randomly (say, i) and replace
             its solution by performing Lévy flights;
          b. Evaluate its quality/fitness Fi [For
             maximization, Fi α f (xi)];
          c. Choose a nest among n (say, j) randomly;
          d. If (Fi>Fj ),
                  i. Replace j by the new solution;
             end if.
          e. A fraction (pa) of the worse nests are
             abandoned and new ones are built;
          f. Keep the best solutions/nests;
          g. Rank the solutions/nests and find the
             current best;
          h. Pass the current best solutions to the next
             generation;
      end while.
```

III. Proposed Approach

In this section, I have provided a brief ideology regarding my proposed workflow:

A. *Pseudo Code*

The following pseudo shows the main working mechanism of the proposed web crawler:

```
    • Variable Initialization:
      url: The address of the website for crawling
      urls: Stack of scrapped urls .
      visited: Historic record of the url.
      tag: To store the filtered links for the best chosen
      solution.
    • Generate an initial population of host nests by storing
      a list of links in the url variable;
    • While (length of url >0)
```

```
        a. Get a cuckoo randomly or say an instance of the input (say, i) and replace its solution by popping out the links from the stack which are already searched using the visited variable;
        b. Evaluate its quality/fitness
            i. Parse the html content gathered from the url and look for a particular tag attribute to pin point the links available in the webpage.
            ii. If (link is not present in the previous visited section) then,
                Store the links in a variable along with its tag name;
                Add the following links to the same variable;
                Normalize the content;
                end if.
        c. Keep the best solutions by storing the filtered links;
        d. Rank the solutions and find the current best by sorting the links as per priority;
        e. Pass current best solutions to the next generation and continue crawling;
    end while.
```

B. Pattern Matching Algorithm

The idea of pattern recognition is used here only for text matching purpose. Knutt-Morris-Pratt (KMP) algorithm [23, 24] works in such a way that pattern and text are put into comparison in a left to right scan. If any similarity is, the algorithm will search for the longest suffix of the "first start" which also serves as prefix of the pattern and thus decides how far the pattern can slide to the right without the consequence of missing a possible similarity. The basic ideology behind KMP's algorithm is: whenever a mismatch is detected (after some matches), already knowing some of the characters in the text (since they were similar with the characters in the pattern prior to the mismatch). Advantage is taken of this information in order to keep away from matching the characters that is known to match anyway. The algorithm performs some preprocessing over the pattern pt[] and builds an auxiliary array lps[] of size m (same as size of pattern). Here, name lps indicates longest proper prefix which is also suffix. For each sub-pattern pt[0…i] where i = 0 to m-1, lps[i] storing the maximum length of the similar prefix which is also a suffix of the sub-pattern pt[0..i].

- Example:

```
Input:   text[] = "AABAACAADAABAAABAA"
         pt[] = "AABA"
Output:  Pattern found in index 0
         Pattern found in index 9
         Pattern found in index 13
```

- **Algorithm:**

Input: pat (pattern text file) string with m number of characters and target web page.

Output: number of comparison the algorithm performs for finding similarity and the time taken to perform the said task.

```
while(j+i < length_of_s):    //character similarity for parallel characters
    a. If (pat[i] == s[j+i]) then: //reached the end of pattern, match found.
        i. If (i == length_of_pat –1) then:
               return j;
           end if.
        ii. i=i+1;    //parallel characters do not match, shift pattern along
    b. else
        i. j=j+i-txt[i];
        ii. If (txt[i] > –1)
                i = txt[i];
            else
                i = 0;
            end if.
    end if
end while.
```

IV. Experimental Analysis

According to the implementation of the crawler, I have seen that this focused crawler saves a lot more time rather than any general web crawler. Thus the focused crawler has a better time complexity than regular crawler but it lacks in the field of space. Speaking of space complexity, the focused crawler utilizes a lot of space as I have to save data at each iteration in order to use it later checking to make the algorithm faster.

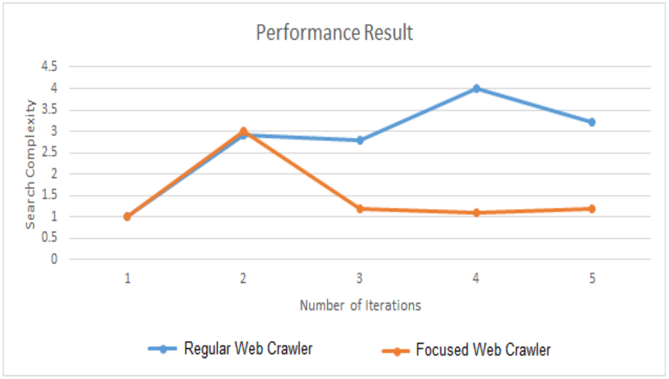

Figure 1. Difference in traffic overhead of regular web crawler and focused web crawler

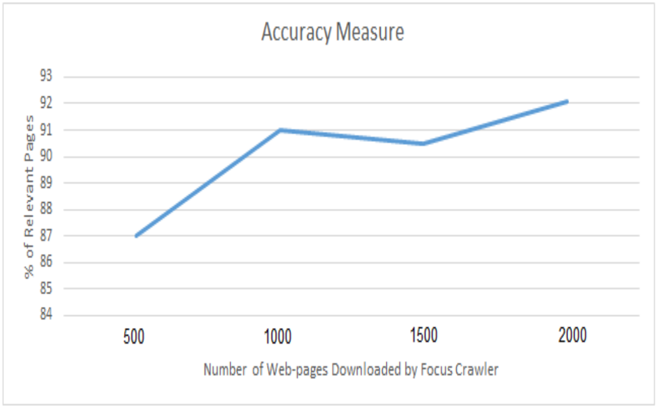

Figure 2. Accuracy measure of the proposed focused web crawler

I was able to reduce the space complexity by setting an overhead limit. The crawler will not search for any particular content for infinite iterations, I can set a limit to how many iterations the algorithm would work for at a particular point of time. As per the Fig. 1 given graph plot, we can clearly see that after a certain point of time the traffic overhead increases rapidly in case of a regular web crawler. This occurs due to overflow of content over the internet and crawler fails to filter its search content from others. But in case of a focused web crawler, the overhead increases initially as it has to seek through various contents and update the data contents, but as the iterations increases, the complexity of search decreases and follows a more sorted fashion. In Fig. 2, I have presented the accuracy percentage of the focused crawler by taking a small volume of data set. However the focus crawler deals with high volume of Web-pages.

V. Conclusion

Thus, in a nutshell, a crawler can be defined as a block of code that downloads web pages, which is mostly done for search engines. The fast growth of World Wide Web poses challenges in searching for a particular link. Focused web crawler is designed to retrieve only the necessary web pages from topics of interest from the Internet. In this paper I used the cuckoo search optimization technique on the focused web crawler. The implemented web crawler is able to formulate a comparison based on the text found in a link with the text file provided as input. This web crawler uses pattern recognition algorithms and formulates repetition of the input text present in the text found on a link. The limitation of this approach is that I have considered only the html web pages.

The crawler performs the pattern recognition using Knutt-Morris-Pratt algorithm in order to normalize the content of the output. The information so produced gives an ideology regarding the accuracy of the pattern matching algorithm. The crawler designed is using only a single type of text mining. The crawler can be further modified to implement other text mining methodology and thus can be used in other efficient techniques and different security features can also be incorporated in the algorithm to provide a security aspect to this algorithm.

References

The following references were considered while working and formulation of this research work:

[1] Yang X., Deb S.: “Cuckoo Search via Levy Flights”. World Congress on Nature & Biologically Inspired Computing, 2009.
[2] Hu K., Wong W.S.: “A Probabilistic Model for Intelligent Web Crawlers”, 27th Annual International Computer Software and Applications Conference.
[3] Sun Y., Councill I. G., Giles C. L.: “The Ethicality of Web Crawlers”, IEEE: International Conference on Web Intelligence and Intelligent Agent Technology, 2010.
[4] Ntoulas A., Cho J, Olston C.: “What's New on the Web? The Evolution of the Web from a Search Engine Perspective”, World-wide-Web Conference (WWW), May 2004.
[5] Arasu A., Cho J., Molina H. G., Paepcke A., Raghavan S.: “Searching The Web”, Computer Science Department, Stanford University.
[6] Cho J., Garcia-Molina H., Page L., “Efficient Crawling Through URL Ordering,” Technical Report, Computer Science Department, Stanford University, Stanford, CA, USA, 1997.
[7] Nath R., Bal S., “A Novel Mobile Crawler System Based on Filtering off Non-Modified Pages for Reducing Load on the Network,” Intenational Arab Journal of Information Technology, Vol. 8, Issue 3, pp.(272-279), 2011.
[8] Shkapenyuk V., Suel T., “Design and Implementation of A High Performance Distributed Web Crawler,” 18th International Conference on Data Engineering, San Jose, CA, IEEE CS Press, pp.(357-368), 2002.
[9] Boldi P., Codenotti B., Santini M., Vigna S., “Ubicrawler: A scalable fully distributed web crawler,” 8th Australian World Wide Web Conference, AUSWEB02, pp.(1-14), Australia, 2002.

[10] Edwards J., McCurley K. S., Tomlin J. A., "An adaptive model for optimizing performance of an incremental web crawler", 10th Conference on World Wide Web, Elsevier Science, pp.(106-113), Hong Kong, 2001.

[11] Najork M., Wiener J. L., "Breadth-first crawling yields high-quality pages", 10th Conference on World Wide Web, Elsevier Science, pp.(114-118), Hong Kong, 2001.

[12] Pinkerton B., "Finding what people want: Experiences with the WebCrawler", 1st World Wide Web Conference, Geneva, Switzerland, 1994.

[13] Chakrabarti S., Berg M., Dom B. E., "Focused Crawling: a New Approach to Topic-specific Web Resource Discovery", 8th International World Wide Web Conference, Elsevier, pp.(545-562), Toronto, Canada, 1999.

[14] Altingovde I. S., Ulusoy O., "Exploiting interclass rules for focused crawling", IEEE Intelligent Systems, Vol. 19, Issue 6, pp.(66-73), DOI: 10.1109/MIS.2004.62, 2004.

[15] Zong X. J., Shen Y., Liao X. X., "Improvement of HITS for topic-specific web crawler", Advances in Intelligent Computing, ICIC 2005, Part I, Lecture Notes in Compter Science, Vol. 3644, pp.(524-532), 2005.

[16] Shivlal Mewada, Sharma Pradeep, Gautam S.S., "Classification of Efficient Symmetric Key Cryptography Algorithms", International Journal of Computer Science and Information Security (IJCSIS) USA, Vol. 14, No. 2, pp (105-110), Feb 2016

[17] Pant G., Srinivasan P., "Link contexts in classifier-guided topical crawlers", IEEE Transactions on Knowledge and Data Engineering, Vol. 18, Issue 1, pp.(107-122), 2006.

[18] Almpanidis G., Kotropoulos C., Pitas I., "Focused crawling using latent semantic indexing-An application for vertical search engines", Research and Advanced Technology for Digital Libraries, Lecture Notes in Computer Science, Vol. 3652, pp.(402-413), 2005.

[19] Diligenti M., Coetzee F., Lawrence S., Giles C. L., Gori M., "Focused crawling using context graphs", 26th International Conference on Very Large Databases, VLDB, Morgan Kaufmann, pp.(527-534), San Francisco, 2000.

[20] Bergmark D., Lagoze C., Sbityakov A., "Focused crawls, tunneling, and digital libraries," European Conference on Digital Libraries, ECDL 2002. Lacture Notes in Computer Science, Roma, Italy, Vol. 2458, pp.(91-106), 2002.

[21] Blum C., Roli A.: Metaheuristics in combinatorial optimization: Overview and conceptural comparision, ACM Comput. Surv, 35, Page No. (268- 308), 2003.

[22] Yang X., "Nature-Inspired Metaheuristic Algorithms". Feb, 2008.

[23] Cormen T. H., Leiserson C. E., Rivest R. L.,: Introduction to Algorithm, Prentice-Hall of India Private Limited, 7th ed, 2009.

[24] Abe U., Brandenburg. :String Matching., Page No (1–9), Sommersemester 2001.

AUTHORS PROFILE

Mr. Joy Dewanjee pursued Bachelor of Technology from Guru Nanak Institute of Technology, West Bengal University of Technology, India in year 2015. He is currently working as an Assitant System Engineer in Tata Consultancy Services. His main research work focuses on Heuristic Algorithms, Web-based optimization techniques, Nucleotide sequencing and alignment in Biotechnology.