A Web Crawler Design for Data Warehousing

Prof. Leena H. Patil[#1], AnkitKhobragade[*2], Priyakant Satpudke[#3], Nikhil Sangani[*4]
Computer Science and Engineering Department,
Priyadarshini Institute of Engineering and Technology- Nagpur, Maharashtra 441110, India

Abstract- *The size of the web is becoming a focus for research activities. The internet is the largest collection of data today. For this computer programs need to conduct any large scale processing of web pages. So we need the use of web crawler at some stage in order to fetch the pages that should be analysed. A web crawler is a program which browses the internet in a methodical, automated manner. This process is called web crawling. A search engine uses web crawler to collect web pages from internet and the web crawler collects it by web crawling. In this paper we have reviewed a web crawler design for data warehousing which allows to search in offline mode also.*

Keywords- *Web crawler; Search engine; Offline mode*

I. INTRODUCTION

The size of the data is increasing on the World Wide Web, it is becoming very important to extract the relevant information in the limited period of time. Many of the research is being done to improve the accuracy of search engines by providing crawling algorithms which could traverse through large number of data in a limited period of time and can return the results which are sorted based.

So, Web crawler is software for downloading pages from the Web automatically. It is also called web spider or web robot. Web crawlers can be used in various areas, the most prominent one is to index a large set of pages and allow other people to search this index[1]. It can be accessed in an offline mode without internet facility.

Search engines use various algorithms which can sort and rank the results in the order of priority to the user's query. Many algorithms are in use - Breadth First Search, Best First Search, Page Rank algorithm, Genetic algorithm to mention a few [2].

The general process that a crawler takes is as follows:- [1]

- It automatically downloads the page we are viewing - the system keeps track of pages to be downloaded in a queue.
- Extract all links from the page and add those to the queue mentioned above to be downloaded later.
- Extract all the words of the page & save them to a database which is associated with this page, and save the order of the words of the page so that people can search for phrases, not just keywords.
- It saves the summary of the page and update the last processed date for the page so that the system knows when it should re-check the page at a later stage.

There are important characteristics of the Web that make crawling very difficult:- [7]

- its large volume
- its fast rate of change, and
- dynamic page generation.

The large volume implies that the crawler can only download a fraction of the Web pages within a given time, so it needs to prioritize its downloads. The high rate of change which implies that the time the crawler is downloading the last pages from a website, it is very likely that new pages have been added to the website, or that pages have already been updated or it have been even deleted.[7]

II. LITERATURE SURVEY

When a data is searched, hundreds of thousands of results appear. Users do not have the persistence and stretch to go through each and every page listed. So search engines have a big job of sorting out the results, in the order of interest to the user within the first page of appearance and a quick summary of the information provided on a page [2].

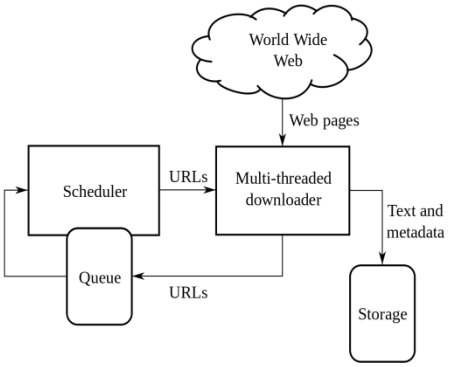

Fig.1 Components of Web Search System [3]

Some of the web crawling algorithms used by crawlers that we will consider are:

- Breadth First Search
- Best First Search
- Fish Search
- A* Algorithm

These three algorithms given are some of the most commonly used algorithms for web crawlers. A* and Adaptive A* Search are the two new algorithms which have been designed to handle this traversal.

A. Breadth First Search

Breadth First Search is the simplest form of crawling algorithm. It starts with a link and keeps on traversing the connected links without taking into consideration any knowledge about the topic. Since it does not take into account the relevancy of the path while traversing, it is also known as the Blind Search Algorithm. It is considered to give lower bound on efficiency for any intelligent traversal algorithm [4].

Pseudo-code for Breadth First Search is as follows:

```
Breadth-First-Search(Graph, root):
    create empty set S
    create empty queue Q

    add root to S
    Q.enqueue(root)

    while Q is not empty:
            current = Q.dequeue()
            if current is the goal:
            return current
            for each node n that is adjacent to current:
            if n is not in S:
                    add n to S
                    Q.enqueue(n)
```

B. Best First Search

Best First Search is a heuristic based search algorithm. In this approach, relevancy calculation is done for each link and the most relevant link, such as one with the highest relevancy value, is fetched from the frontier [5]. Thus every time the best available link is opened and traversed.

Pseudo-code for Best First Search is as follows:

```
function best_first(start, finish)
closed = set({})
open = set({start})

score_of = {}
score_of[start] = calculate_heuristics(start)

whilenotopen.is_emptydo
-- find node with minimum f
current = min(open, function(node) returnscore_of[node] end)
if current == goal then
--reconstruct the path to goal
returncreate_path(current)
end
closed.add(current)
open.remove(current)

forneighborincurrent:neighbors() do
ifnotclosed.contains(neighbor) then
--calculate the heuristics for neighboring node
score_of[node] = calculate_heuristics(neighbor)
--add neighboring node to open set
open.add(neighbor)
end
end
end

-- there is no path to the goal
end
```

C. Fish Search

Fish Search is a dynamic heuristic search algorithm. It works on the intuition that relevant links have relevant neighbours; hence it starts with a relevant link and goes deep under that link and stops searching under the links

that are irrelevant. The key point of Fish Search algorithm lies in the maintenance of URL order.[8]

Pseudo-code for Best First Search is as follows:[8]

1. Initialize user parameters
2. Initialize fishes positions randomly
3. while Stopping condition is not met do
4. Calculate fitness for each fish
5. Run individual operator movement
6. Calculate fitness for each fish
7. Run feeding operator
8. Run collective-instinctive movement operator
9. Run collective-volitive movement operator
10. end while

D. A* Algorithm

A* algorithm combines all the features of uniform-cost search and pure heuristic search to efficiently compute optimal solutions. A* algorithm is the Best First Search algorithm in which 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 node n and h(n) is the heuristic estimate of the cost of the path from node n to the goal node. Thus, f(n) estimates the lowest total cost of any solution path going through node n. At each point a node with lowest f value is chosen for expansion. It ties among nodes of equal f value should be broken which is in favour of nodes with lower h values. The algorithm terminates when a goal node is chosen for expansion [6].

III. PROPOSED METHODOLOGY

Breadth First Search is a blind search algorithm and hence not a very efficient method. Best First Search and A* Search show nearly equal search time and can be improved using better heuristic functions. Work can be done on improving the heuristic function in Best First Search and A* Search so as to increase the efficiency of the algorithms, the accuracy and timeliness of search engines. Better A* Search will result in the improvement of all other search, by which we can hope to make Web crawling much faster and more accurate. Further improvement can be made by dynamically updating the heuristic approach based on the traversal done in reaching the intermediate node from the initial node, thus improving the remaining part of the journey using the enhanced heuristic function.A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals. It is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently. It is really a smart algorithm which separates it from the other conventional algorithms.

Pseudo-code for A* Approach is as follows:[6]

```
/*Start with given Seed URLs as input*/
A_Star_Algo(Initial seed)

/*Insert Seed URLs into the Frontier*/
Insert_Frontier (Initial seed);

/*Crawling Loop*/
While (Frontier! = Empty)

        /*Pick new link from the Frontier*/
        Link: = Remove_Frontier (URL);

        Webpage: = Fetch (Link);

        Repeat For (each child_node of Webpage)

                /*Calculate Relevancy Value till that Page*/
                Rel_val_gn (child_node):= Rel_val(topic, node webpage);

                /*Calculate Relevancy Value from that Node till the Goal Page*/
                Rel_val_hn (child_node):=Rel_val(topic, goal webpage)- Rel_val(topic, node webpage);

                /*Calculate Total Relevancy Value of the Path to the Goal Page*/
                Rel_val_fn:= Rel_val_gn+Rel_val_hn;

                /*Add new link with Maximum Relevancy Value into Frontier*/
                Insert_Frontier (child_node_max, Rel_val_max);

End While Loop
```

IV. CONCLUSION

The paper surveys several crawling methods or algorithms that are used for downloading the web pages from the internet. Web crawlers are an important aspect of all the search engines. They are the basic component of all the web services so they need to provide high performance. A number of crawling algorithms are used by the search engines. A good crawling algorithm should be implemented for better results and high performance. In this paper we have studied different crawling technologies where how to crawl searching a hidden web documents with different ways. We have tried to make our working design as simple as possible. Compared to other crawling technology the focused and hidden web crawling technology is designed for advanced web users on the particular topics. Our future work will be including a complete implementation, analysis and evaluation of this approach. In future, work can be done on improving the heuristic function in Best First Search and A* Search so as to increase the efficiency of the algorithms, the accuracy and timeliness of search engines. Better A* Search will result in the improvement of Adaptive A* Search, by which we can hope to make Web crawling much faster and more accurate

V. REFERENCES

[1] Mini Singh Ahuja Dr Jatinder Singh BalVarnica, *"Web Crawler: Extracting the Web Data"*, International Journal of Computer Trends and Technology (IJCTT) – volume 13 number 3 – July 2014

[2] Pavalam, S. M., SV Kashmir Raja, Felix K. Akorli, and M. Jawahar, *"A Survey of Web Crawler Algorithms*," International Journal of Computer Science, vol. 8, iss. 6, no 1, Nov. 2011.

[3] Component of web search system figure, Accessed July 25,2017. https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwionMns_aTVAhXLpo8KHaB0BPQQjRwIBw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FWeb_crawler&psig=AFQjCNGV10hs3Dm9EBk_yJgFmXskFXXq6g&ust=1501090566478463

[4] http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/11-Graph/bfs.html

[5] https://www.researchgate.net/figure/315347498_fig1_Fig2-Pseudocode-for-Best-First-Search-algorithm

[6] http://db.cs.duke.edu/courses/fall11/cps149s/notes/a_star.pdf

[7] Shalini Sharma,"Web Crawler", International Journal of Advanced Research in Computer Science and Software Engineering- Volume 4, Issue 4, April 2014

[8] Aviral Nigam, "Web Crawling Algorithms", International Journal of Computer Science and Artificial Intelligence Sept. 2014, Vol. 4 Iss. 3, PP. 63-67