Web Crawling Algorithms

Aviral Nigam

Computer Science and Engineering Department,
National Institute of Technology - Calicut, Kozhikode, Kerala 673601, India
aviral@nitc.ac.in

***Abstract*-As the size of the Internet is growing rapidly, it has become important to make the search for content faster and more accurate. Without efficient search engines, it would be impossible to get accurate results. To overcome this problem, software called "Web Crawler" is applied which uses various kinds of algorithms to achieve the goal. These algorithms use various kinds of heuristic functions to increase efficiency of the crawlers. A* and Adaptive A* Search are some of the best path finding algorithms. A* uses a Best-First Search and finds the least-cost path from a given initial node to a goal node. In this work, a study has been done on some of the existing Web Crawler algorithms and A*/Adaptive A* methods have been modified to be used in this domain. A*/Adaptive A* methods being heuristic approaches can be used to find desired results in web-like weighted environments. We create a virtual web environment using graphs and compare the time taken to search the desired node from any random node amongst various web crawling algorithms.**

Keywords- Web Crawler; A*; Adaptive A*

I. INTRODUCTION

With the amount of data increasing on the World Wide Web, it becomes extremely important to extract the most relevant information in the shortest span of time. A lot of research is being done to improve the efficiency of search engines by providing crawling algorithms which could traverse through large chunks of data in a short span of time and return the results sorted based on their relevance.

Search engines use algorithms which can sort and rank the results in the order of proximity to the user's query. Many algorithms are in use - Breadth First Search, Best First Search, Page Rank algorithm, Genetic algorithm, Naive Bayes classification algorithm to mention a few [1].

Whatever information we get might not be completely useful. There might be a scenario where the website might contain a large number of popularly searched keywords, just to increase the number of hits on their website. So there is always the challenging task to get search engines which provides relevancy, robustness and the ability to download optimal number of pages.

With heuristic approach being compared to native techniques of web crawling, we focus on a comparative study between these approaches. The comparison will be done on simulated web environment and the time to search is the time taken to reach the most relevant page from any initial page.

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 [1].

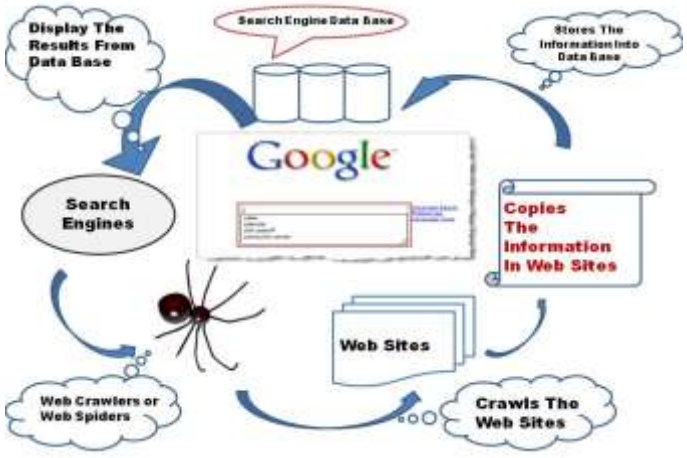

Fig.1Components of a Web Search System with Web Crawler [2]

DOI: 10.5963/IJCSAI0403001

Retrieving effective content from the Web is a crucial task because it heavily influences the perceived effectiveness of a search engine. Users often look at only a few top hits, making the precision achieved by the ranking algorithm of paramount importance. Early search engines ranked pages principally based on their lexical similarity to the query. The key strategy was to devise the best weighting algorithm to represent Web pages and query in a vector space, so that closeness in such a space would be correlated with semantic relevance [3].

Web Crawler is a program/software or automated script which browses the World Wide Web in a methodical, automated manner [4]. Crawlers have bots that fetch new and recently changed websites, and then indexes them. By this process billions of websites are crawled and indexed using algorithms (which are usually well-guarded secrets) depending on a number of factors. Several commercial search engines change the factors often to improve the search engines process [1].

The basic procedure executed by any web crawling algorithm takes a list of seed URLs as its input and repeatedly executes the following steps [3]:

- Remove a URL from the URL list.
- Download the corresponding page.
- Check the Relevancy of the page.
- Extract any links contained in it.
- Add these links back to the URL list.
- After all URLs are processed, return the most relevant page.

A* algorithm combines 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. Ties among nodes of equal f value should be broken in favour of nodes with lower h values. The algorithm terminates when a goal node is chosen for expansion [5].

Adaptive A* uses A* Search to find shortest paths repeatedly. It uses its experience with earlier searches in the sequence to speed up the current A* Search and run faster than Repeated Forward A* [6]. In a given state space with positive action costs, the task of Adaptive A* is to repeatedly find cost-minimal paths to a given set of goal states. The searches can differ in their start states. Also, the action costs of an arbitrary number of actions can increase between searches by arbitrary amounts. Adaptive A* uses informed h-values to focus its searches. The initial h-values are provided by the user and must be consistent with the initial action costs. Adaptive A* updates its h-values after each search to make them more informed and focus its searches even better [7].

III. WEB CRAWLING ALGORITHMS DESIGN

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

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

The first 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 [8].

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 [9]. Thus every time the best available link is opened and traversed.

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

DOI: 10.5963/IJCSAI0403001

point of Fish Search algorithm lies in the maintenance of URL order.

D. A Search*

A* uses Best First Search. It calculates the relevancy of each link and the difference between expected relevancy of the goal web-page and the current link. The sums of these two values serve as the measure for selecting the best path.

Pseudo-code for A* Approach is as follows:

```
/*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
```

E. Adaptive A Search*

Adaptive A* Search works on informed heuristics to focus its searches. With its each iteration, it updates the relevancy value of the page and uses it for the next traversal. The pages are updated for log(Graph Size) times, (after log(Graph Size) times the overhead of updating is much more than the improvement that can be achieved in getting more relevant pages) and then normal A* traversal is done.

Pseudo-code for Adaptive A* Approach is as follows:

```
/*Start with given initial Seed URL as input*/
Adaptive_A_Star_Algo(Initial seed, Graph Size)

/*No. of times relevancy has to be updated to get better results*/
b: = log(Graph Size);

Repeat For (b times)

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

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


DOI: 10.5963/IJCSAI0403001

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

Webpage: = Fetch (Link);

Repeat For (each child_node of Webpage)

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

        /*CalculateRelevancyValuefrom 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

/*After b times, A* Search more efficient on updated graphs*/
A_Star_Algo(seed URL, Graph (G));
```

IV. DESIGN IMPLEMENTATION

In the implementation of these algorithms, we have made some assumptions related to Relevancy Calculation and Link Extraction. To calculate the relevancy of different documents, random relevancy values between 0 and 1 are generated. Whenever a page is extracted, a random relevancy is generated corresponding to it. All Link extractions are done by traversing a graph and maintaining lists of unvisited pages.

Since the Web is a directed graphical structure, for implementation purpose a graph is considered where each page is represented by a node and link from one page to another is represented by an edge. In the implementation process, web crawler is provided with a seed URL(Graph Node) and then it uses the algorithm to traverse the graph. The time taken for each traversal is the difference between the start time and end time of each algorithm.

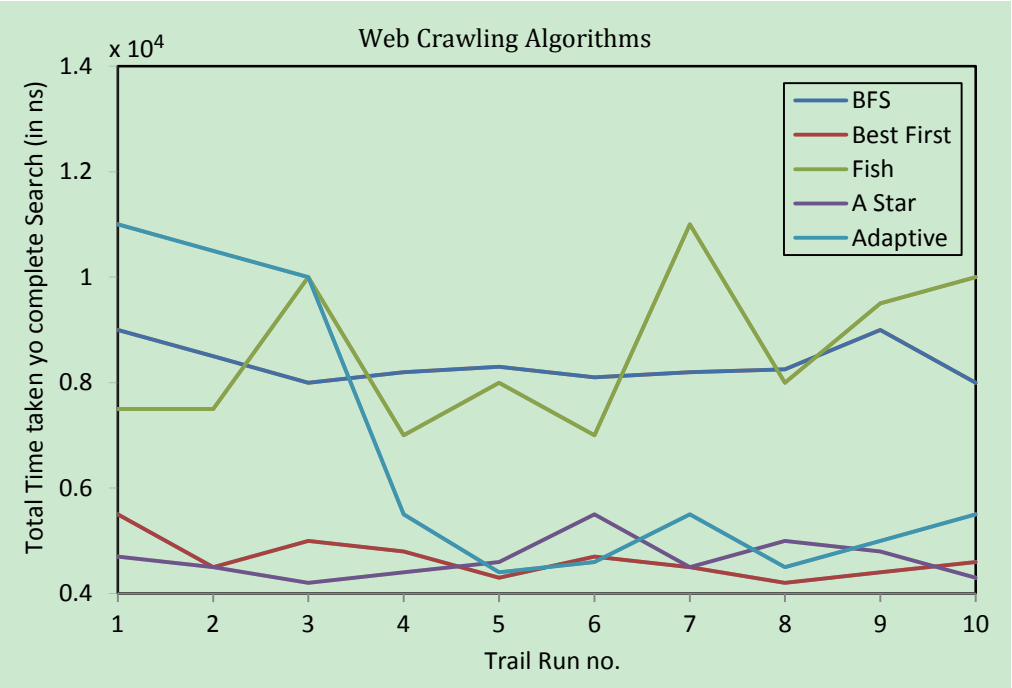

Fig.2 Search Time of Various Web Crawling Algorithms

DOI: 10.5963/IJCSAI0403001

V. RESULT AND ANALYSIS

The graph above, Fig. 2; shows an approximate trend in the running time of these algorithms. The actual running time largely depends on the initial node, the relevancy of the nodes and the network structure.

Though the graph may not be completely accurate, it shows the general trend found while running the algorithms with different initial nodes. The X - axis represents the trail number, such as the particular instance in the sequence of trails and Y - axis represents the total time taken in nanoseconds to complete the journey from the initial node to the goal node. Even though the difference in total time taken to search might look large from the graph, the actual difference is very less as all the calculations are done in nanoseconds.

The general trend observed is in terms of the starting node; only Fish Search is largely dependent on the initial node, all the other algorithms show nearly the same result with different initial nodes.

In terms of search time; Breadth First Search takes the maximum amount of time as it blindly searches through all nodes irrespective of the initial node. Fish Search shows unpredictable trend as its search time highly depends on the initial node and since it requires large amount of code execution, it might give more comparable results for large graphs. Both Best First Search and A* Search can be said to give the best results as they require lesser time than other algorithms and are also fairly constant in their travel time. Adaptive A* takes more time initially, but once it gathers appropriate amount of knowledge about the graph it takes the nearly same time as the Best First Search and A* Search, and also provides the most relevant information.

VI. CONCLUSION

Reducing the search time with relevant results is one of the major problems faced by search engines these days. There are a large number of algorithms used by Web Crawlers to increase their efficiency and the approaches discussed in this report are also a part of that group.

Breadth First Search is a blind search algorithm and hence not a very efficient method. Fish Search has been facing criticism due to its property of searching down a particular path and ignoring elsewhere and has been modified to Shark Search to get better results. Best First Search and A* Search show nearly equal search time and can be improved using better heuristic functions. Since a lot of users search same type of content, using Adaptive A* Search will prove efficient as it stores the history of previous searches and with every search, the efficiency of search will increase. Therefore, running an Adaptive A* Search will give more relevant results even though initially it may consume some amount of time.

The algorithm can work efficiently in static as well as less dynamic environments where environment variables do not change during the search. If the environment drastically changes during the search then the current process will not be able to produce efficient result. This is one of the major weaknesses in the current algorithm that does not allow dynamic change in heuristic approach based on drastic changes in the environment.

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. 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.

REFERENCES

[1] 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.

[2] Web Crawler Figure, Accessed March 30, 2013. http://onesourcegraphics.org/wp-content/uploads/2012/01/Presentation1.jpg.

[3] Menczer, Filippo, Gautam Pant, and Padmini Srinivasan, "Topical webcrawlers: Evaluating adaptive algorithms," ACM Transactions on Internet Technology (TOIT), vol. 4, no. 4, pp. 378-419, 2004.

[4] Sharma, Sandeep, and Ravinder Kumar, "Web-Crawling Approaches in Search Engines," Technical Report, Thapar University, Patiala, 2008.

[5] A* Search, Accessed Feburary 24, 2013, en.wikipedia.org/wiki/A*_search_algorithm.

[6] Koenig, Sven, and William Yeoh, A project on fast trajectory replanning for computer games for 'introduction to artificial intelligence' classes, Technical report, Department of Computer Science, University of Southern California, Los Angeles (California), 2008.

[7] Sven Koenig and Maxim Likhachev, "Adaptive A*," In Proceedings of the International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), pp. 1311-1312, 2005.

[8] Breadth First Search, Accessed March 16, 2013, en.wikipedia.org/wiki/Breadth-first_search.

[9] Best First Search, Accessed March 14, 2013, en.wikipedia.org/wiki/Best-first_search.

[10] Sandeep Sharma and RavinderKumar, "Web-Crawlers and Recent Crawling Approaches," International Conference on Challenges and Development (IT-ICCDIT-2008), PCTE, Ludhiana (Punjab), May 30th, 2008.

DOI: 10.5963/IJCSAI0403001