Page Quality Optimization in Crawler's Queue through Employing Graph Traversal Algorithms

Saeideh Tajbar-Parashkoohi
Department of Software Engineerig, Islamic Azad University, Rasht Branch, Rasht, Iran

Fatemeh Ahmadi-Abkenari
Department of Software Enginnering and Information System, Payame Nour University (PNU), Iran

ABSTRACT
In today's information era, Web becomes one of the most powerful and fastest means of communication and interaction among human beings. Search engines as Web based applications traverse the Web automatically and receive the set of existing fresh and up-to-date documents. The process of receiving, storing, categorizing and ndexing is done automatically based on partial smart algorithms. Although many facts about the structure of these applications remains hidden as commercial secrets, the literature tries to find the best approaches for each modules in the structure of search engines. Due to the limited time of today's Web surfers, providing the most related and freshest documents to them is the most significant challenge for search engines. To do so, every module in search engine architecture should be designed as smart as possible to yield not only the most related documents but also to act in a timely manner. Among these modules is the sensitive part of crawler. One of the open issues in optimization of search engines' performance is to reconfigure crawling policy in a way that it follows the most promising out-links that carries the content related to the source page. Crawler module has the responsibility to fetch pages for ranking modules. If higher quality pages with less content drift are indexed by the crawlers, the ranking module will perform faster.

According to the graph structure of the Web, the way of traversing the Web is based on the literature on graph search methods. This paper experimentally employs different graph search methods and different combinations of them by issuing some queries to Google engine to measure the quality of received pages with fixing the factor of graph depth to identify the best method with reasonable time and space complexity to be employed in crawler section in search engine architecture.

Keywords
Graph Traversal approaches, Search Engine Optimization (SEO), Web Crawler, Web Page Ranking Methods.

1. INTRODUCTION
Along with significant growth of the World Wide Web and regarding the dynamic nature of the Web, providing the most accurate search results is of the main demands of search engines' users. Web crawler is a section of a search engine that by starting from some seed pages traverses the Web graph and stores the primary set of addresses in a queue while stores the Web pages content in the search engine repository. Web crawler obtains the next addresses to follow from the out-links of the downloaded pages and puts new addresses in the starting point queue and obtains next address from this queue. Web crawler repeats the crawling process until the stop decision is made. Web crawler often downloads millions of pages in short period of time, monitors them continuously and updates its address queue. Additionally, Web crawler should respect the server rules by following the robot.txt file and should try to avoid overloading Web servers [1], [8], [12], [16].

Most Web crawlers have five following main sectors:

1. Decision maker module
2. Fetching module
3. Control module
4. Filter module
5. Workload module

Decision maker module (DNS) searches for IP addresses in the domain names. This module determines where the determined page should be fetched. The compiler which is monitored by the control unit goes to seed pages and sends the documents to the filter unit which uses the HTTP protocol to restore pages. Text filter module extracts a series of the links from fetched pages. After separation, suitable links are sent to workload unit and put in next instruction list for the fetching unit. Actually, the filter unit includes two sectors, link filtering and indexing. Figure 1 represents the main functions of the Web crawler [4], [6], [15], [17].

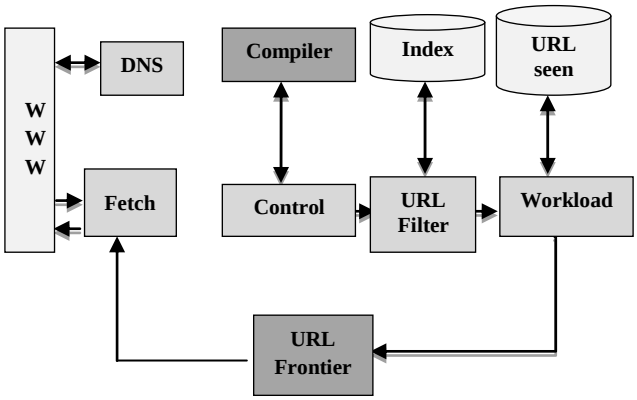

Fig 1: Web crawler architecture

Since the best search engines have a database of about fifty percent of the Web pages, identifying more important links is much critical in search engines efficiency in order to provide user's satisfaction. In fact when a user requests a query, instead of sending the query to millions of Web sites, it is compared with a list of pre-processed data in order to find the best match. Preprocessing is done by crawlers that perform the extraction of Web pages in order to analyze and create index on a regular, rapid and comprehensive routine and to deliver them to pages storage. Considering the large number of Web pages, the crawler can only download a fraction of them which cannot be selected randomly. Therefore, it is essential that the Web crawler be smart enough to be able to download pages in order of their priority and importance. Thus, defining the behavior of a Web crawler is a

combination of strategies in order to select an algorithm for making decision on downloading pages with higher priority, revisiting Web pages for updating and avoiding the overload on Web servers.

In this paper, first, various graphs traversal methods are reviewed in order to identify the best method to use in a Web crawler module. The advantages and disadvantages of them are analyzed. Then, based on the conducted experiments, the best graph traversal method with the possibility of applying in decision-maker module of the Web crawlers is selected and introduced to recognize the efficiency of link importance that prevents or lessen the content drift.

2. PROBLEM STATEMENT

The crawler section of a search engine is not a smart module. It traverses the Web, follows any out-links and download Web pages in order to make a repository of reasonable dimension. In search engines, the responsibility of making decision on the content authority of the pages and their quality regarding different issued queries is on the shoulder of ranking and analyzer modules. These modules in a tedious and offline process, checks the quality of pages and index them according to various content categories. If the crawler section does the crawling process in a smart manner and instead of blindly following out-link, follows the more promising links with less content drift from the seed page, crawling queue will be full of higher quality pages and the overall performance of search engine will be optimized in a more timely and intelligent manner. To do so, this paper aims at employing a conducted experiment on using different graph traversal algorithms by issuing different queries in order to check the quality of resulted pages at leaf level regarding their less content drift to the seed page.

3. GRAPH TRAVERSAL STRATEGIES

Web structure can be considered as a huge oriented graph which contains nodes as Web pages and multiple connections and links among pages as edges. Crawling strategies of the Web can be classified in three general categories, uninformed/blind search, informed/heuristic search and local search.

3.1 Uninformed Search

In uninformed search strategy, the only existing information defines the problem and the target state against non-target state. Here there is no idea about how and in what paths the target should be reached. As a result in blind methods the overall search space has been traversed for finding the target. Some methods like depth-first search (DFS), breadth first search (BFS) and uninformed cost search (UCS) are examples of uninformed search algorithms.

Depth first search (DFS) algorithm was introduced in 1994 and was applied in Web crawler as the best algorithm for many years. The crawlers based on this search method follow the links using a frontier as FIFO queue. Control unit of the crawler determines a page as a starting point page for fetching unit. After filtering links, control unit selects one of external links of the page and introduces destination node to the fetching unit. Movement process among the pages continues until interested depth level is faced. When a node and its offspring have been extended in that path, they are removed from memory. Thus, this method have a linear memory and space complexity of $O(bm)$ in which b is the branching factor or the maximum number of edges that goes out of a node and m is the depth of the tree. In worst case, this method extends all nodes of the search graph in which the time complexity will be $O(b^m)$. Difficulty with high time complexity could be faced in the indefinite path that the solution could not be reached because of the false primary option damaging the perfectness of the method. Against the fascinating simple nature of this algorithm, many low quality pages regarding their irrelevant content to the source page will be stored in repository [7], [11].

In breadth first search (BFS) method, after specifying the seed page, the control unit determines all nodes with same breadth and introduces them to the fetching unit. After the crawler visits all of the pages specified in that level, the unit control goes to second breadth and reviews it. This method always has been drawn attention in crawler literature and for software designers because of its easier designing and implementation. In this method because of the limitation in the number of out-links, the size of repository will not increase impracticably. In depth-first search, fetching unit reviews all links in a page up to the defined depth and goes to the next page so receives more information about an special subject. Figure 2 shows the pseudo code of the breadth first search algorithm [5], [11].

```
Breadth-First (starting_urls) {
        foreach links  (starting_urls)  {
            Enqueue (frontier, link);
        }
        While (visited < MAX_PAGES) {
             Link: = dequeue_link (frontier);
             Doc: = fetch (link);
             Enqueue (frontier, extract_links(doc) );
             if (#frontier > MAX_BUFFER)    {
                  dequeue_last_links (frontier);
             }
        }
   }
```

Fig 2: A crawler with breadth first search strategy

Since complexity of this method is modal and high, this method is not efficient. Assuming branching factor as b and in worse case, target in depth of d, it extends all of nodes in depth d. The number of expanded nodes up to the specified depth is $1+b+b^2+\ldots+b^d$. When reaching d breadth, number of expanded nodes is b^d and therefore time complexity is $O(b^d)$. Each node that expanded in this method should be saved in the memory, as a part to generate other nodes. Thus the space complexity of this method is same as its time complexity [13], [14].

The optimized algorithm as uninformed cost search (UCS) is based on breadth-first method in which the first node to expand is the node with minimal cost. To implement this traversal method, a priority based queue is employed. The remarkable point about uniform cost search is that it finds optimal solution if the costs of each step is properly elected. The problem with this method is the development of additional nodes that slow down finding the solution. The time and space complexity of this method is $O(b^{\lceil c^*/\varepsilon \rceil})$ in which c^* is the estimated cost of the optimal path and ε is the cost of each step.

3.2 Informed/Heuristic Search

In blind search methods, in the worst case, all nodes in the state space should be inspected to reach the solutions, however, if the number of nodes is very large, the underlying methods do not reach to the intended target in a reasonable time. To solve this problem, informed search methods are

developed. In these methods, in addition to defining the problem, solutions are provided to achieve the target. In other words, there is information about which none-target states are more appropriate than others. In fact, the target in the informed search is to find ways by which only a subset of nodes is expanded instead of traversing all nodes in the stated space. For this reason, the search strategy in the methods using a heuristic function *f(n)* will expand the best node at each step. So it can be concluded that informed search methods include two general parts, the search strategy and the function heuristic. Some of the most important informed search strategies are best first search and A* search algorithms [22].

There are various algorithms derived from the best-first method definition which is employed in shark search, focused crawlers, information spiders and so on. In the best-first method, a page *A* makes out-link to another page *B* if *B* is beneficial in *A*'s point of view. One of the measures in being best is the employment of page ranking algorithms that based on them, the control unit selects the pages according to the rank of each page and sends them to the fetching unit [2], [3], [5]. The crawlers using best-first method considers a page as beneficial if:

- More links point to that page. Establishing more links to a page shows importance of the page and its authority.
- Receiving links from more authorized pages lifts the authority of the page itself [9], [10].

The rank of a page could be calculated via the formula (1) as follows [20];

$$\mathrm{R(U)} = \sum_{v \in b_u}^{n} \frac{R(V)}{|F_U|} \qquad (1)$$

In formula (1), main page indicator is *U*, *Fu* indicates those pages receive links from the page U and *Bu* indicat links to the page *U*. *R(V)* is the rank of the pages est links to the page *U*. Links are selected by simple calculations using lexical similarity between keywords and source pages. Therefore, the similarity between a page *P* and the keywords is applied to estimate the relation of pages that were referenced by *P* and an URL is chosen with the best estimation for crawling. Also, the *Cosine* similarity is used and the links with the lowest similarity scores are removed from this range [18], [19].

Function *Sim()* in formula (2) returns *Cosine* similarity between the query and the page in which *q* is the interested query, *p* is the fetched page and f_{kp} is the frequency of term *k* in *p*.

$$Sim\ (q\ ,\ p) = \frac{\sum_{k \in q \cap p} f_{kq} f_{kp}}{\sqrt{\left(\sum_{k \in p} f_{kp}^2\right)\left(\sum_{k \in q} f_{kq}^2\right)}} \qquad (2)$$

The pseudo code of the crawler which applies a best-first strategy is as shown in figure 3;

```
BestFirst (topic, starting_urls) {
    foreach links (starting_urls) {
        Enqueue (frontier, link);
    }
    While (visited < MAX_PAGES) {
        Link := dequeue_top_link(frontier);
        Doc := fetch(link);
        Score := sin(topic, doc);
        Enqueue (frontier, extract_links(doc), score);
        if (#frontier > MAX_BUFFER) {
            dequeue_botton_links(frontier) ;
        }
```

Fig 3: A crawler with best first search strategy

The best-known form of best-first search is A* search that is one of the most complex search algorithms. A* search method tries to keep minimum the total paid cost to the current node and the remaining cost from current node to the target. Estimation of remaining cost up to the target is known as problem heuristic. Heuristic design of the problem in A* search method is important and the optimization level of A* method is highly affected with the problem heuristic. The heuristics that does not meet this requirement is called unacceptable heuristic. The evaluator function of A* method is *f(n)=g(n)+h(n).* In fact, *f(n)* is the estimated cost of the cheapest solution through *n*. According to above, it could be stated that *F* is total cost of the node, *G* is cost to current node and *H* is estimated cost to target node. In the A* search method, the next node to expand is a node with the lowest cost *F* among other unexpanded nodes. In the A* method, always in an operating environment the leaf nodes that have not been expanded are developed and studied. So all nodes are saved in memory and some nodes may frequently be assessed and evaluated [22], [23]. Therefore, the time and space complexity of this method is modal and high as $O(b^m)$. Najork and Weiner (2001) demonstrated in practice that considering high cost and time of the ranking Web pages and the instability nature of ranking process, the breadth-first method acts better than the ranking method.

3.3 Local Search

Search algorithms that have been explained so far, are designed in such a way that one or more paths are kept in memory until the target is found. The path to that target will be chosen as the solution. But in many issues, the path to the target is not important but achieving the target is significant. In these problems, another set of search algorithms under the name of local search is used. In these methods, rather than considering multiple paths, decision and action are only based on the current state. In this kind of algorithm, after moving from the current state to another, the state will be shifted to its neighbors. The paths are taken in the search, are not saved in memory. Local search algorithms, in addition to finding the target are suitable for solving optimization problems where the objective is to find the best state based on an objective function. Hill climbing search, local beam search, simulated annealing search (SA) and threshold acceptance algorithm (TA) are some of the most important local algorithms.

In hill-climbing algorithm (HC) at first, a solution to the problem is generated randomly and then in a loop manner until the stop condition of the algorithm has not been established, a number of neighbors of the current state are generated frequently. Among neighbor, the best is chosen and replaced by the current state. Implementation of the hill

climbing method requires two functions, objective and neighbor functions. Objective function determines the optimization level of the solution and neighbor function generates current state neighbors. The following pseudo-code in figure 4 shows the algorithm of the hill climbing algorithm [21];

```
Procedure HillClimbing
    Generate a solution (S')
    Best = S'
    Loop
    S = Best
    S' = Neighbors(S)
    Best = SelectBest (S')
    Until stop criterion satisfied
End
```

Fig 4: A crawler with hill climbing search strategy

Another search method based on hill-climbing search is called local beam search that has much power than the hill-climbing search in resolving the problems. In this method, unlike the hill-climbing method, at first, *K* solutions is generated to the problem. Then, for each of the *K* state, it produces its neighbors and among all neighbors, *K* neighbors are selected as best neighbors that the process continues until achieving the stop condition. Choosing *K* more efficient solutions among all generated neighbors solution prevents the solutions similarity to each other. Local beam search algorithm is shown in figure 5;

```
Procedure LoaclBeamSearch
  Generate K solution
  Do
    For each solution generate its neighbors
    Select K best solution from whole neighbors
    Replace current solutions by selected solutions
  Loop until stop criterion satisfied
End
```

Fig 5: A crawler local beam search strategy

Simulated annealing search (SA) method is another search method that its idea has been originated from gradual cooling of metals in order to strengthening them more. As in hill-climbing method, in this method, the problem is started from a state space like *S*. By transition from one state to another, it closes to the problem optimal solution. The starting state selection can be done randomly and can be chosen based on the initial state rule. Objective function calculates the optimization level of the current state and neighbor generates the neighbor state to the current state. It is important how to generate neighbor state. The general method is that in each iteration, the SA algorithm generates a neighbor state like *S'* based on a probability, the problem goes from state *S* to state *S'* or stays within the same state *S*. This process is repeated until a relatively optimal solution is obtained or the maximum number of iterations is reached. In this algorithm, it was stated that generated neighbor state acceptance is done based on a probability. Function *P(e,e',T)* determines the probability of acceptance of neighbor state. Optimization level of current state is *e* and optimization level of neighbor state is *e'*. If the neighbor state is worse than the current state, the parameter *T* determines the probability of solution acceptance [21]. At the beginning, the value *T* has been chosen so that most of neighbor states are accepted. The parameter *T* is the temperature indicator and the value of this parameter is gradually reduced. Parameter value *T* is chosen so that before the maximum number of iterations, its value becomes almost zero. The evidences for the SA algorithm show that, at first it is better to determine the value of *T* such that 80% of the solutions will be accepted by the algorithm. Simulated annealing search algorithm is shown in figure 6;

```
Procedure Simulated Annealing
  C = Choose an initial solution
  T = Choose an initial temperature
  REPEAT
    S' = Generate a neighbor of the solution C
    ΔE = objective( S' ) – objective( C )
    IF (ΔE > 0 ) THEN // S' better than C
      C = S'
    ELSE with probability EXP( ΔE/ T )
      C = S'
END IF
```

Fig 6: A crawler local beam search strategy

Threshold Acceptance algorithm (TA) method is like SA method with the only difference in the acceptance of non-optimal solutions.

TA algorithm accepts the solutions that are not much worse than the previous solutions. Like what temperature does in SA algorithm, temperature in the algorithm must be chosen in such a way that most optimal solutions initially are accepted by the algorithm.

4. EXPERIMENTAL RESULTS

As stated before regarding the large number of Web pages, the crawler should download a fraction of them that this fraction selection could not be done randomly. So, the crawler should be smart enough to be able to download the most promising pages in order of priority and importance regarding their content. The importance diagnosis algorithms are the responsibility of the crawler decision maker module. In this section the carried out test is described with the aim of selecting a comprehensive algorithm to detect importance of the links in order to download the fraction of Web pages with least content drift regarding the source node. In this test, the depth-first methods, breadth-first, hybrid of the depth-first and breadth-first methods in three forms of H1, H2 and H3, best-first and hill climbing (HC) algorithm have been evaluated by issuing different queries. For representing the results of issued queries five of them have been selected as Q1: "Computer networks", Q2: "Artificial Intelligence", Q3: "Web crawler", Q4: "Cloud Computing" and Q5: "Search engine". While the obtained results from other queries support the results of these five queries that are described in continue. The queries have been issued to Google search engine and then the quality of result set has been evaluated to determine the best traversing algorithm. Due to non-optimal feature of other search methods regarding their high time and space complexity, employing of them is avoided in this test. Also, in some search methods always a node is considered as the target node that this assumption could not be used in Web traversal. Such algorithms are uniform cost search (UCS), A*, local beam search, simulated annealing search and Threshold Acceptance method that are omitted from the test.

The CPU used to carry out the test is Intel core i7, Q720 1.60 GHz with 4GB of Memory. The crawl has been done in time period from June 29th 2014 till September 10th 2014. The number of crawled and analyzed Web pages for each of queries in different search methods has been shown in table 1.

The limited number of crawled pages in best first search algorithm originated from the nature of this method that specifies the authority of pages and then follows the authorized links.

Table 1. Number of crawled and analyzed Web pages for each of queries in different traversal algorithms

	BFS	DFS	H1	H2	H3	Best	HC
Q1	4056	6081	6450	5910	5910	300	2217
Q2	3890	4975	5512	4782	4125	246	1758
Q3	1970	2730	3100	2800	2500	170	900
Q4	3200	4505	4824	3917	3754	235	1475
Q5	2476	2970	3300	2940	2754	220	1100

First of all, the seed pages must be selected at first. Our measure for selecting seed pages is the number of related external links. After the search on first query ("Computer networks") through reviewing all pages, three pages were selected as seed pages as S_1, S_2 and S_3 that have more number of associated external links. Then the test continues as described below:

In the breadth first method, all the external out-links related to the first seed page are extracted and called $a_1, a_2, \ldots, a_n$. At this breadth considering five levels the result of 90/90% is obtained. At the second breadth, all external out links from $a_1, a_2, \ldots, a_n$ is extracted and called $b_1, b_2, \ldots, b_n$ in a way that $a_1b_1, \ldots, a_1b_n, a_2b_1, \ldots, a_2b_n, \ldots a_nb_1, \ldots a_nb_n$. At this breadth, the relevance percentages of the pages are calculated for each b_i and the relevance of pages are obtained as 33.83%. In the third breadth, as well as other breadths, the external out- links from $b_1, b_2, \ldots, b_n$ are addressed as $c_1, c_2, \ldots, c_n$ respectively. At this breadth, the number of relevant pages is reduced and is dropped to zero. In next breadths of fourth and fifth that called $d_1, d_2, d_3, \ldots, d_n$ and $e_1, e_2, e_3, \ldots, e_n$, third breadth result is repeated. The traversed path has been depicted in figure 7. Therefore, the average of total relevant pages for S_1, S_2 and S_3 using breadth first method is 33.16% . The content relevance percentage in BFS method for the five queries has been depicted as the first column in figures 9 to 13. The same path is traversed for other queries in breadth first manner.

In experimenting with the depth-first method, first the search depth is fixed at 5. In most search engines, the depth level is defined by the control unit. After searching and examining up to the fifth depth, the results of this method is compared with the breadth-first method In this method, first starting from the seed pages one by one and considering their first out-link as *a1*, its relevance to the interested content of *S1* is examined. Then going to the second depth, first the out links of a_i as b_i is extracted. Then out-links extracted from each b_i is c_i and the relevance of c_i to the interested content is calculated. In the fourth and fifth depths, the action is like the previous depths, as these extracted out- links is called d_i and e_i. Finally the relevance average of the five depths is calculated. The first step of the test is shown in figure.

Level 0	S_1, S_2, S_3
Level 1	$(S_1 . a_1 \rightarrow \ldots \rightarrow S_1 . a_n)$
Level 2	$(S_1.a_1.b_1 \rightarrow \ldots \rightarrow S_1.a_1.b_n) + \ldots + (S .a_n.b_1 \rightarrow . \ldots \rightarrow S_1 . a_n.b_n)$
Level 3	$(S_1 . a_1.b_1 .c_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_1 . c_n) + (S_1 . a_1.b_2 .c_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_2 . c_n) + \ldots \rightarrow (S_1 . a_1.b_n .c_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_n . c_n) + (S_1 . a_2.b_1 .c_1 \rightarrow \ldots \rightarrow S_1 .a_2.b_n. c_n)$
Level 4	$(S_1.a_1.b_1 .c_1.d_1 \rightarrow \ldots \rightarrow S_1.a_1.b_1 .c_1 .d_n) + (S_1.a_1.b_1 .c_2. d_1 \rightarrow \ldots \rightarrow S_1.a_1.b_1.c_2.d_n) + \ldots \rightarrow (S_1 . a_1.b_1.c_n .d_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_1 .c_n.d_n) + (S_1. a_1.b_2 .c_1. d_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_n. c_n. d_n) + \ldots$
Level 5	$(S_1.a_1.b_1 .c_1.d_1 .e_1 \rightarrow \ldots \rightarrow S_1.a_1.b_1 .c_1 .d_1.e_n) + (S_1.a_1.b_1 .c_2. d_2.e_1 \rightarrow \ldots \rightarrow S_1.a_1.b_1.c_1.d_2.e_n) + \ldots \rightarrow (S_1. a_1.b_1.c_1 .d_n .e_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_1 .c_1.d_n.e_n) + (S_1. a_1.b_1 .c_2. d_1.e_1 \rightarrow \ldots \rightarrow S_1 .a_1.b_n. c_n. d_n.e_n) + \ldots$

Fig 7: The traversed path in breadth first method for five levels.

$$S_1\, a_1 \rightarrow S_1a_1\, b_1 \rightarrow S_1a_1b_1\, c_1 \rightarrow S_1a_1b_1c_1\, d_1 \rightarrow S_1a_1b_1c_1\, d_1$$

Fig 8: Beginning of the path in depth first method

Considering figure 8, the relevance average of pages for sees page of S_1, is 10.08 %. In this test it can be seen that in the depth-first-movement, a more and unnecessary detailed view of the content is provided to the user. By comparing the results of employing the two methods of BFS and DFS started from S_1 related to query of "Computer networks", it can be seen that in the breadth-first the relevance of the pages is more than depth first method. By observing the obtained values, it can be concluded that in all levels of S_1, the breath-first method acts better than the depth-first method so the user faces more relevant pages. The content relevance percentage in DFS method for the five queries has been depicted as the second column in figures 9 to 13.

In this paper, in order to achieve better and more related and authorized results, a hybrid method of breadth first and depth first is used by following each seed pages. This test is done in three different combination levels in a way that in the first combination (H1) form, the breadth-first for one level is implemented and for other four levels the DFS is employed. In the second combination (H2) form, both first and second breadths are traversed as BFS while third, fourth and fifth levels are traversed as DFS. In third combination (H3) form, the first, second and third breadths are traversed by BFS and fourth and fifth levels by DFS.

Tables 2 and 3 show the content relevance of pages using any of BFS, DFS, and hybrid method and the best-first and hill climbing methods for all tested queries and figures 9 to 13 shows the relevance percentage of the result set for each of the queries with BFS, DFS, H1, H2, H3 and hill climbing.

As depicted in figures 9 to 13, the best result is related to the best first search method. But considering the space and time complexity of this method and the tedious process of filtering and determining the most authorized pages, this method could not be applied in search engines. The second best result according to the conducted experiment and based on the results of searching different queries belongs to the first hybrid method of H1. Since the basis of this method is the simple DFS and BFS methods, it could be applied in crawler module to fill the crawler queue with pages with higher quality.

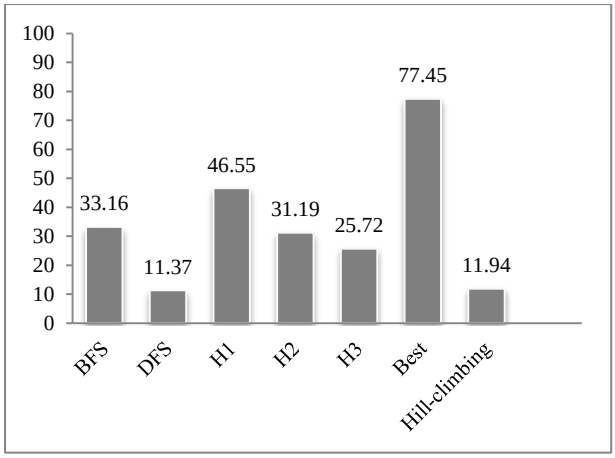

Fig 9: Relevance percentage of the pages in "Computer networks" query (Q1)

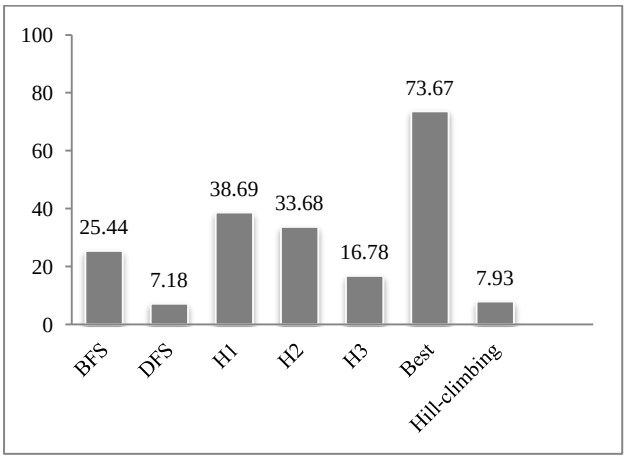

Fig 10: Relevance percentage of the pages in "Artificial Intelligence" query (Q2)

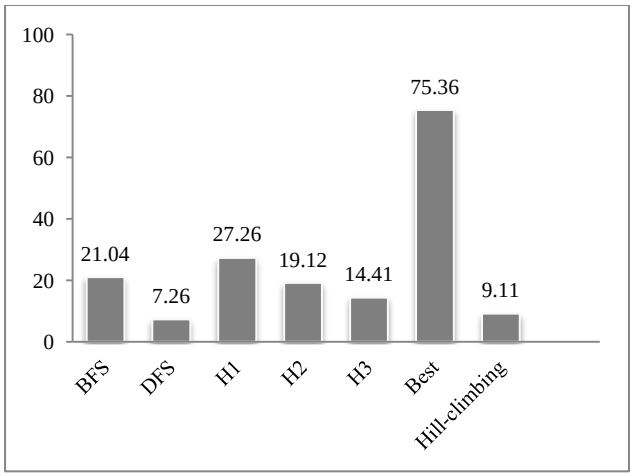

Fig 11: Relevance percentage of the pages in "Web crawler" query (Q3)

Table 2. The relevance of pages in search methods of BFS, DFS, best first and hill climbing

	BFS	DFS	Best First Search	Hill-Climbing
Q_1	33.16	11.37	77.45	11.94
Q_2	25.44	7.18	73.76	7.93
Q_3	21.04	7.26	75.36	9.11
Q_4	24.02	11.03	72.85	13.47
Q_5	18.68	8.17	70.28	11.25

Table 3. The relevance of pages in hybrid search methods

	Hybrid (1)	Hybrid (2)	Hybrid (3)
Q_1	46.55	31.19	25.72
Q_2	38.69	33.68	16.78
Q_3	27.26	19.12	14.14
Q_4	39.95	37.82	21.01
Q_5	30.64	29.47	13.98

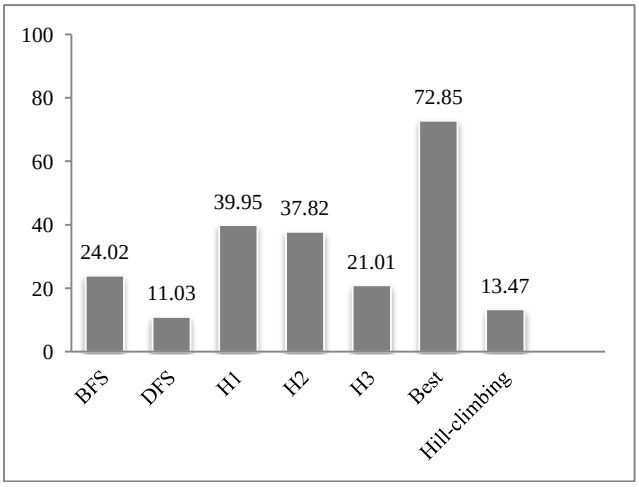

Fig 12: Relevance percentage of the pages in "Search engine" query (Q4)

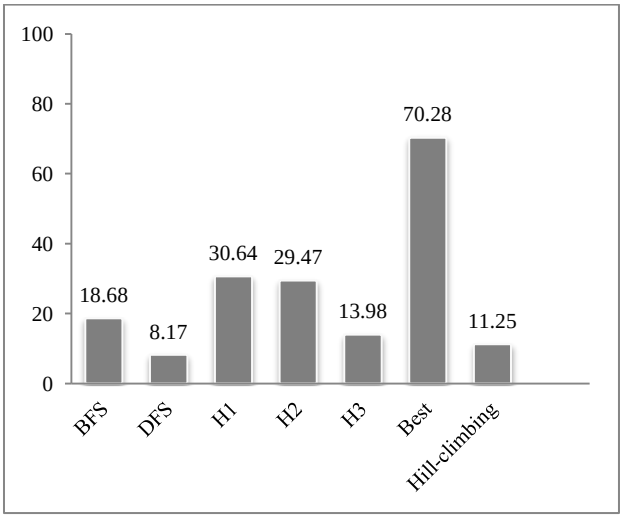

Fig 13: Relevance percentage of the pages in "Cloud Computing" query (Q5)

5. CONCLUSION

In order to have a crawler queue with pages of higher quality regarding the authority of Web pages and no or less content drift to the seed page, search engines should employ more intelligent crawler module. This paper uses different graph traversal approaches and hybrid methods based on them to yield the result set for issued queries. Then the quality of obtained pages has been checked regarding the level of content drift. Based on our experimental results, the best first search method is the search algorithm which produces the highest quality pages. But regarding its time, space and cost complexity, the second best result which is the hybrid level one of breadth first and depth first methods is proposed as the best applicable search algorithm for search engines. The future work of this paper seeks an optimized algorithm based on best first search method in combination with the hybrid level one in some extent to produce better quality pages in comparison to the findings of this paper.

6. REFERENCES

[1] Ahmadi-Abkenari, F., Selamat, A. 2012. "An Architecture for a Focused Trend Parallel Web Crawler with the Application of Clickstream Analysis", International Journal of Information Sciences, Elsevier, Vol. 184, pp. 266-281.

[2] Ahmadi-Abkenari, F., and Selamat, A. 2013. "Advantages of Employing LogRank Web Page Importance Metric in Domain Specific Web Search Engines". JDCTA: International Journal of Digital Content Technology and its Applications. Vol. 7, No. 9. pp. 425-432.

[3] Ahmadi-Abkenari, F., and Selamat, A. 2012. "LogRank: A Clickstream-based Web Page Importance Metric for Web Crawlers". JDCTA: International Journal of Digital Content Technology and its Applications. Vol. 6, No.1. pp. 200-207.

[4] Arastoo poor ,sh. 2008. "The Crawler and Web structure" information and library journal, Vol. 9, No. 2, pp. 4-15.

[5] Baeza-Yates R., Castillo C., Marin M., and Rodriguez A. 2005. "Crawling a country: Better strategies than breadth-first for Web page ordering". In Proceedings of the 14th international conference on World Wide Web / Industrial and Practical Experience Track, Chiba, Japan,. ACM Press, pp. 864– 872.

[6] Esmaeeli, m. tavakoli,m, hashemi majd, s, 2013. "The Web crawler" APA professional laboratory in context of information and communication technology security, document number, APA_FUM_W_WEB_0111, pp. 5-28, bahman.

[7] Hafri Y, and Djeraba C. 2004. "High performance Crawling system". In Proceedings of the 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval pp. 299–306.

[8] Junghoo Cho, 2002. "Parallel Crawlers". In proceedings of WWW2002, Honolulu, Hawaii, USA, May 7-11. ACM 1-58113-449-5/02/005.

[9] Junghoo Cho, Hector Garcia-Molina, and Lawrence. 1998. "Efficient Crawling through URL Ordering Page". In Proceedings of the 7th World-Wide Web Conference. pp. 161-171.

[10] Kumar G., Duhan N., and Sharma A.K. 2011. "Page Ranking Based on Number of Visits of Links of Web Page". International Conference on Computer & Communication Technology (ICCCT)-2011, IEEE, pp. 11-14.

[11] MENCZER F and SRINIVASAN P. 2004. "Topical Web Crawlers: Evaluating Adaptive Algorithms", ACM Transactions on Internet Technology'. Vol. 4, No. 4, pp. 378–419.

[12] MENCZER, F., PANT, G., RUIZ, M., AND SRINIVASAN, P. 2001. "Evaluating topic-driven Web Crawlers". In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel, Eds. ACM Press, New York, NY, pp .241–249.

[13] M. Kurant, A. Markopoulou, and P. Thiran. 2010. "On the bias of BFS (Breadth First Search)". In arXiv: 1004. 1729.

[14] Najork, M., Wiener, J.L. 2001. "Breadth-First Search Crawling Yields High-Quality Pages". In WWW'01,10th International World Wide Web Conference. pp. 114-118.

[15] Olston Ch, and Najork M. 2010. "Web Crawling'. Foundations and Trends in Information Retrieval". Vol. 4, No. 3, pp .175–246.

[16] Onn Brandman, Junghoo Cho, and Hector Garcia-Molina. 2000. "Crawler Friendly Servers". In Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS). Santa Clara, California.

[17] Pant G. and Menczer F. 2003. "Topical Crawling for Business Intelligence". In Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003), Norway.

[18] Pant G., Srinivasan P., and Menczer F. 2002. "Exploration versus Exploitation in Topic driven Crawlers". In WWW02 Workshop on Web Dynamics, Hawaii.

[19] Pant G., Srinivasan P., and Menczer F. 2004. "Crawling the Web". Web Dynamics, pp. 153-178.

[20] Tyagi N., and Sharma S. 2012. "Weighted Page Rank Algorithm Based on Number of Visits of Links of Web Page". International Journal of Soft Computing and Engineering (IJSCE) , Vol. 2, Issue-3.

[21] Hoffmann, J. 2000. "A heuristic for Domain Independent Planning, and its Use in an Enforced Hill-Climbing Algorithm". 12th International Symposium on Methodologies for Intelligent Systems (ISMIS-00), Springer, pp. 216–227. Berlin.

[22] Stern, R., Kulberis, T and Felner, A. 2010. "Using Lookaheads with Optimal Best-First Search". Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10). pp. 185-90.

[23] Reid, M and Korf, R.E. 1998. "Complexity Analysis of Admissible Heuristic Search". American Association for Artificial Intelligence (AAAI-98), pp. 1-6.