RESEARCH REVIEW JOURNALS
Volume-04
Issue-02
February -2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com [UGC Listed Journal]

Web Crawling Model and Architecture

[1]Dr Anish Gupta, [2]Dr. K. B. Singh and [3]Dr. R. K. Singh

[1]Department Of Information Technology, B. R. Ambedhar Bihar University, Muzaffarpur
[2]PG Department of Physics, Samastipur College, Samastipur, LNMU, Darbhanga
[3]Department of EC and IT, MIT Muzaffarpur, Bihar

ARTICLE DETAILS

Article History
Published Online: 20 February 2019

Keywords
web crawler, network, search engine, software architecture.

ABSTRACT

Web crawlers [1] concurrently contend with multiple problems that can even contradict each other. To re-visit pages together with finding new pages concurrently, fresh copies of the web pages are held [3]. Usage of limited infrastructure such as network connectivity without web servers being overwhelmed. They ought to have many "good pages" but they don't specify exactly which pages are the preferred alternatives in advance.

This paper explores a paradigm that directly combines crawling with all existing search engine and, using customizable criteria, retrieves potential answers to cope with conflicting objectives [2]. We explain how this model generalises several individual cases, and add a crawling programming structure which really utilizes the model.

1. The problem of crawler scheduling

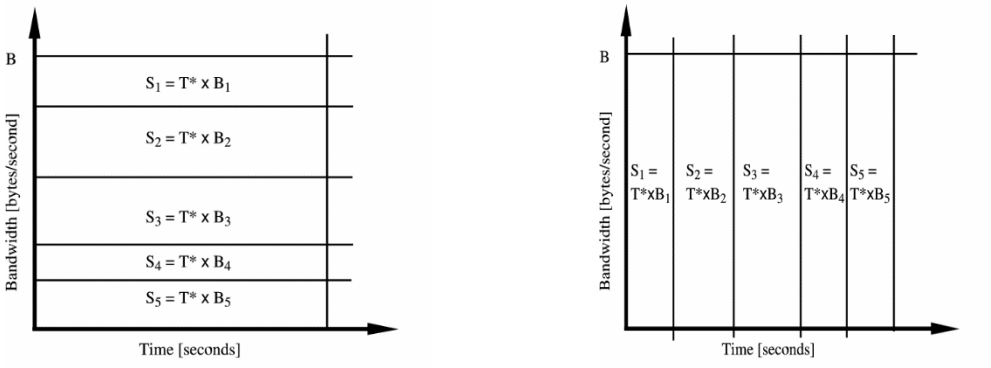
Using a bandwidth, W, of a network connection measured in bytes/second, we envision a search engine that must download a set of pages where size of each page *p*, X_p in bytes. The purpose of the crawler is to download in the shortest period of time all the pages. A trivial approach to this problem is to concurrently import total pages and use a portion of the space on each page equitable to the height of each page. $W_p$is the downloading speed for page *p*, then:

$$W_p = \frac{X_p}{T^*} \quad (1.1)$$

Where N* is the optimal time to use all of the available bandwidth:

$$N^* = \frac{\Sigma_p X_p}{W} \quad (1.2)$$

This scenario is depicted in Figure 1.1a.



(a) Full parallelization (b) Full serialization

Figure 1.1: Two hypothetical Information retrieval scenarios for all page download : (a) parallelizing and (b) serialising . Page sizes are represented by the regions, as size = speed*time.

Many limitations, however, prohibit this ambitious outcome. The primary limitation is to prevent overloading websites by planning policies and maintaining a culture of politeness. Until rendering a submission, a web crawler can wait for several seconds. It should ensure that only one page should be accessed at a time.

We should serialise all requests instead of copying several web pages at a time, i.e. downloading one page at a time with full speed, as shown in Figure 1.1b. The bandwidth required for W_3^{Max} websites, however, is normally smaller than the crawler bandwidth W, so this example is also not possible. The findings provided imply that the real download time lines are identical to the time lines seen in Figure 1.2. The optimum time N* is not reached in the figure because some bandwidth is lost due to constraints

in the speed of websites (the maximum speed for page three is seen in the figure, W_3^{Max}) and the fact that the crawler has to wait between visiting a website.

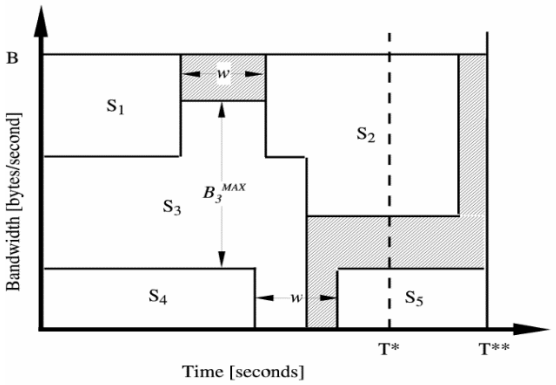


Figure 1.2: A more practical time line for Web crawlers for downloading. Pages one-two and four-five belong to the same place, and the crawler waits between them for a few seconds. Owing to the weaknesses of the scheduling policy, the hatched component is unused bandwidth. It does not reach the optimum time N*.

We should try to use the complete link to import pages via multiple websites in order to solve the problems shown in Figure 1.2. The maximum number of pages is always accessible on some small sites: the distribution of pages to sites, as seen in Figure 1.3, is very low in terms of crawler scalability. This limits robotics' high performance and optimal usage of bandwidth.

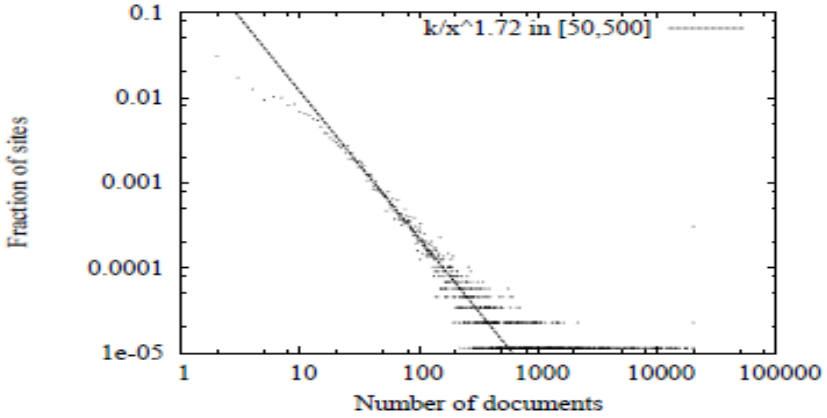


Figure 1.3: Distribution of site sizes within the Chilean Page sample. There are several websites that are not tall, and the full size of the web page is too little. As they have to download full pages from a limited number of websites, this is one of the web crawler's greatest obstacles. There are a few very big websites and a large fraction of small websites, but they have to ensure at the same time that the request does not flood them, resulting in disappointment.

There is another important technical constraint: there is latency in the HTTP request, and the latency time would be more than 25% of the overall request time [7]. This latency is basically the time it takes to create the TCP link, and if the same connection is used to issue several requests using the HTTP/1.1 'keep-alive' function, it may be partly overcome.

2. Issues in a Searching Model

Crawling literature deals specifically with the words "crawler" and "spider" and walks through a graph of these words. Crawling is an automated page uploading mechanism that does not follow any browsing pattern: it follows the first wide approach in a few situations. The standard algorithm for crawling comes from the World Wide Web's early days.

Batch indexing is executed, multiple megabytes of text at a time, and it is very costly to do it one document at a time using current algorithms, unless an exact balance can be found between the incoming document stream and the index processing speed [8], and in this case the index construction becomes the bottleneck. Expertise from the creep approaches to the database to be indexed. Thus, the index is not constantly but completely changed at the same time in many search engines. This is also the pattern for several leading search engines applicable to the literature, and there is even a term coined for upgrading Google's database ('Google dance'), when the new index is circulated to the various data centers [9]. When the index is updated in batches, which URLs were first transferred is meaningless.

In batches, dispersed crawlers swap URLs. If the crawler is used in a distributed manner, the crawler must return the results to a central server or share the results with other processes of crawling. It must send several URLs at a time for better results, as the exchange of URLs produces an overhead that is mainly provided by context switches, not by the (relatively small) scale of the URLs [4]. This means that the global crawling order does not change how the URLs are locally organised.

Earlier in the crawl, the essential URLs are shown. If any URL ordering is achieved and if this ordering is not based on a query's text similarity, then a page we have just seen is a very unlikely candidate to be downloaded in the near future in a steady state: "good" results are seen soon [5]. Alternatively, in a late stage of the crawling process, if a URL is used for the first time, there is a high chance that it is not a really interesting website. If Pagerank [2] is used, this is clearly valid because it represents the time a random surfer spends on the page and if a random surfer spends more time on a page, then it is possible that the page will be accessed from many connections.

We also found that two related problems appear to be distinguished by previous work and to merge two distinct problems:

- Short-term performance, optimal bandwidth utilization and being polite with servers, and long-term efficiency, prioritise the important page first, are the two separate issues that are typically combined. In Section 1, we address why these two topics should be distinguished.
- The freshness of the index and the underlying accuracy of the index are the two associated topics usually known as separate issues. We assume that in terms of a list of scores, such as freshness, it is better to think of the different features of the documents in the collection, which should be weighted according to certain targets that vary depending on the essence of the use of the crawler. This conception is further developed in Section 2.

3. Separating short-term from long-term scheduling

We plan to deal with long-term scheduling and short-term scheduling separately. So, we need to show that all topics can be segregated, namely, we need to verify if the inherent output of a web page or a web server is linked to the bandwidth available to access the page. If that is the case, so it would not be feasible to pick the most relevant pages first and later re-order pages to use the bandwidth effectively, because we would affect the speed of network communication when choosing the important pages.

We planned and ran the following experiment to test this hypothesis. We took one thousand Chilean site names at random from the roughly 50,000 currently present. We repeatedly visited the homepage of these websites every 6 hours over a span of 2 weeks and calculated the communication speed (bytes/second) and latency of these websites (seconds). In order to avoid interruption arising from discrepancies in links to different servers, sites were accessed sequentially to get a measure of the network transmission characteristics (not in parallel).

We were able to calculate 750 of them effectively from the 1000 home pages, as the others were down for a large fraction of the observed time, or did not address our request for an individual web page. We only consider the websites that responded to the requests in the study.

We also used the number of in-links from various websites on the Chilean Web as an indicator of the "importance" of websites, as this is a comparative measure of the visibility of the website by other website owners.

The correlation coefficient r was determined between the number of in-links and the velocity (r = −0.001) and the number of in-links and the latency (r = −0.069). The correlation is not statistically important between these parameters. These findings indicate that for long-term scheduling, the variations between network links to "important" pages and "normal" pages are not significant. The scatter plot of the linking speed and number of in-links as seen in Figure 1.4.

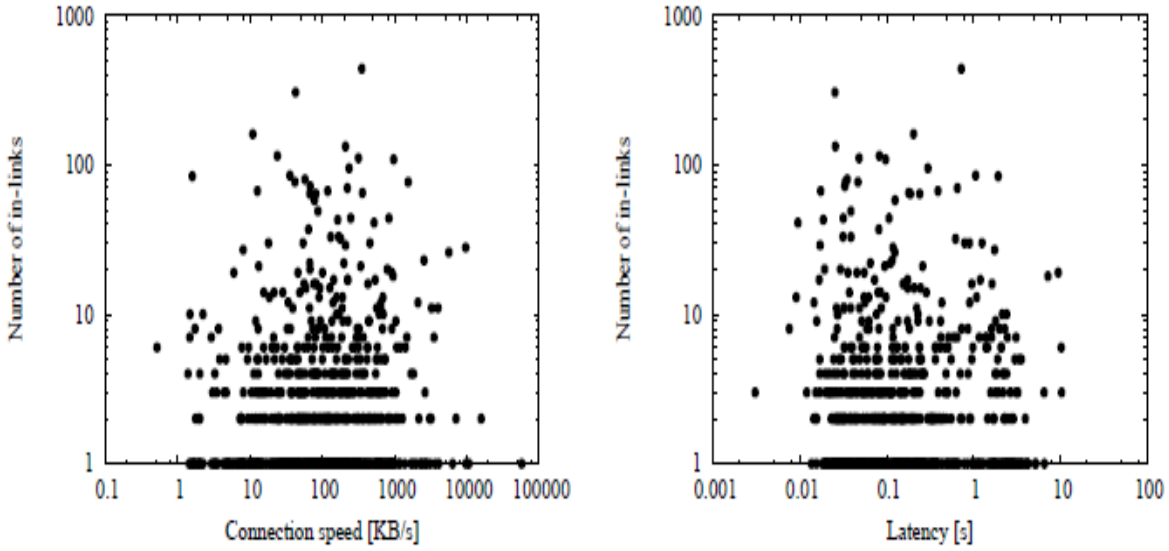


Figure 1.4: Link speed (in Kilobytes per second) and latency (in seconds) versus popularity scatter plot (measured as the number of in-links). We observed no relevant association between these variables in our experiments. The findings are an estimate of the measurements obtained by sequentially linking to 750 websites every 6 hours over a 2-week period.

We have used the data obtained during this experiment to calculate the association between the connection speed and latency (r = -0.645), which is strong, as seen in Figure 3.5., for completeness. We can see in the graph that higher-bandwidth websites appear to have low latency times.

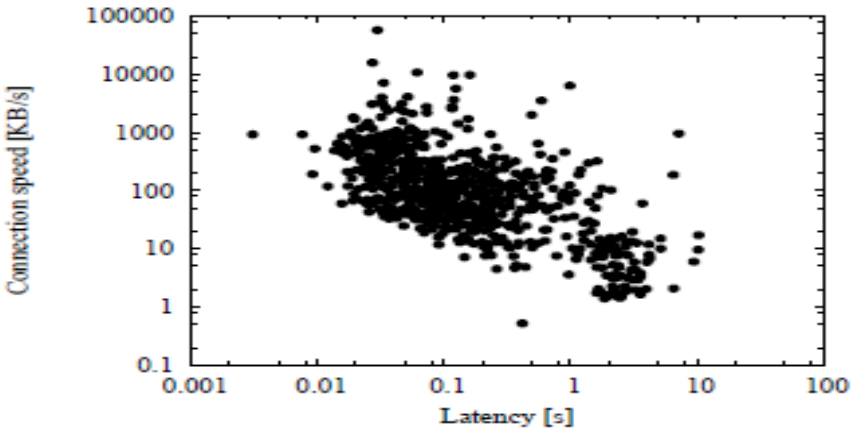


Figure 1.5: Scatter plot of velocity versus latency relation. Web pages with low contact rates tend to have low latency as well.

Another important finding we observed from this test was that over the observed era, connection speeds and latency times differed considerably. On average, we find a relative deviation of 42 percent for velocity and 96 percent for latency, but these two quantities can not be predicted only on the basis of their mean values observed. For a more detailed forecast, the regular and weekly periodicity in Web server response time found by Liu must be taken into consideration: Diligentiet al.[10] keeps multiple observed values for link speed prediction, and groups the measurements by time of day to account for the periodicity in Web server response time.

4. A software architecture

To build a new crawling architecture, the observations provided in the previous sections can be used. The aim of the design of this crawling architecture is to split the crawling mission into multiple tasks that specialised modules can carry out effectively.

With two units, as seen in Figure 1.6, a division of tasks can be accomplished. The scheduler measures scores and assigns pages to several downloader modules that pass pages, parse their contents, retrieve new links and preserve the configuration of the connection across the network.

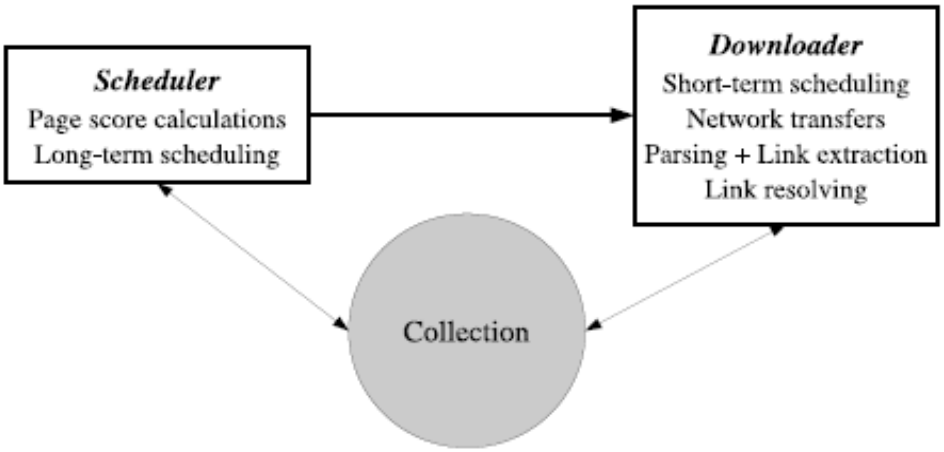


Figure 1.6: A two-module device architecture. The scheduler creates a "batch" of pages, and the downloader downloads and parses them. Under this scheme, read access to the set is required by the scheduler and read and write access by the downloader.

This two-module design has several issues. One concern is that the Web graph can not adjust during the calculations for the scheduler to run on the Web graph. The method of updating the Web graph could then be as simple as possible, but it can be sluggish to parse the pages and this may mean that we have to "lock" the Web graph for a long time. To solve this, what can be done is to parse all pages and accumulate links, and then add to the list all the links found.

Another challenge is that, for the tasks of uploading and saving pages and for the task of parsing pages, we should have separate, streamlined hardware architectures. In terms of processing, parsing pages can be costly, whereas uploading pages requires a high degree of network access and fast discs. In addition, if it is appropriate to carry out network downloads with high parallelism, then each download activity should be very light. We split down the tasks of copying, decoding, and preserving the relation structure, as seen in Figure 1.7, to solve these problems. In the thesis, the following module names are used:

Manager: page value calculations and long-term scheduling.

Harvester: short-term scheduling and network transfers.

Gatherer: parsing and link extraction.

Seeder: URL resolving and link structure.

Figure 1.8 introduces the main data structures that form the index of the search engine, and outlines the steps of the operation:

1. **Efficient crawling order** The "manager" module, which produces the list of URLs that the harvester can download in the next period (a "batch"), performs long-term scheduling. The goal of this module is to optimise the "profit" in each loop (i.e.: to increase the index value).

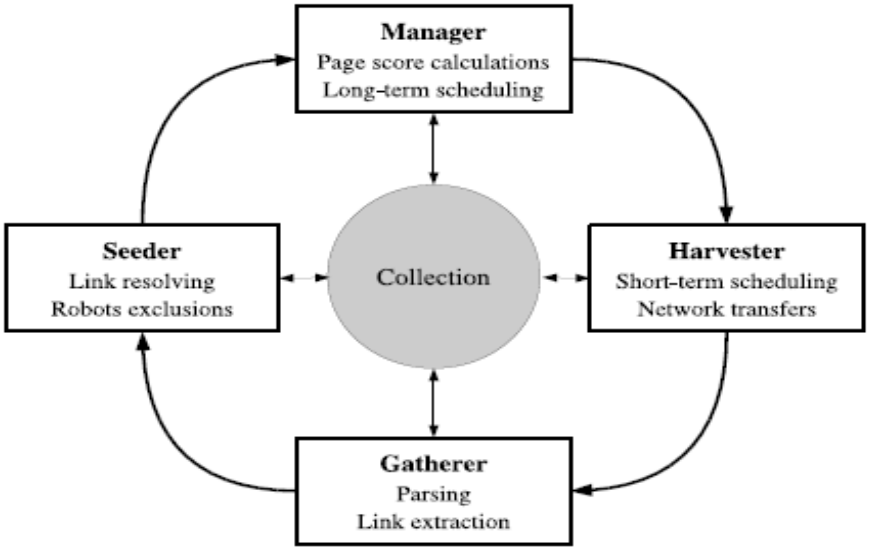


Figure 1.7: A manager is presented with the planned programme architecture, which creates loads of URLs to be downloaded by the harvester. Then the pages go to a gatherer that parses them and sends the URLs found to a seeder.

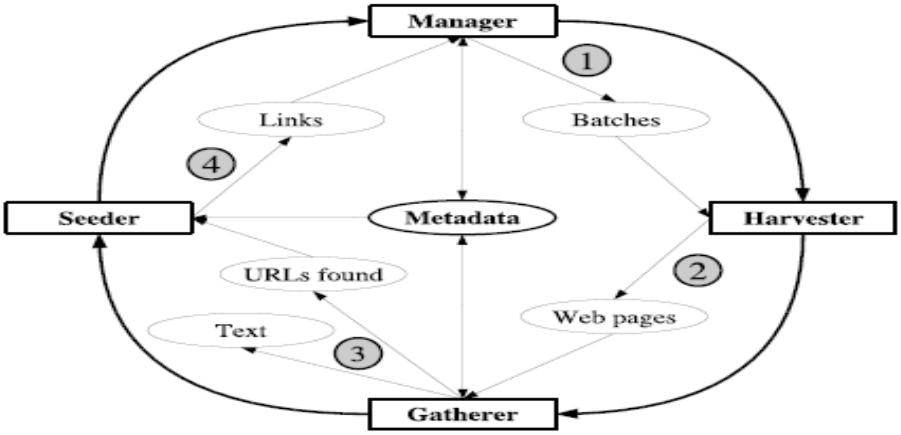


Figure 1.8: The main data structures and the operation steps of the crawler: (1) the manager generates a batch of URLs, (2) the harvester downloads the pages, (3) the gatherer parses the pages to extract text and links, (4) seeder checks for new URLs and maintains the link structure.

2. **Effective network transfers** The 'harvester' module is delegated to short-term scheduling. This module gets loads of URLs and its goal is to download as easily as possible the pages in the batch, use different links and implement a policy of politeness. A partial array, consisting mainly of raw HTML data, is created by the harvester.

3. **Effective page parsing** The "gatherer" module is allocated to retrieve the text and links. The partial collections downloaded by the harvester(s) are obtained by this module and the text is applied to the main collection. A list of found URLs that are passed to the seeder is also created.
4. **Effective manipulation of URLs** The URLs find are processed by a seeder module that looks for new URLs not seen before. This module also scans for URLs that can not be browsed due to the exclusion protocol of robots.txt. A data structure documenting Web connection is maintained by the module.

The pattern of data structure read and write access is intended to increase the crawler's scalability as, for example, the pages can be accessed and parsed while the site graph is evaluated, and the analysis must stop only while the seeder is running. In Figure 1.8, the systems and data structures.

5. Conclusions

Network crawling is not just a trivial problem in graph traversal. This includes many challenges that result from the Web's distributed existence. First, web crawlers must share resources with other agents, often of humans, and should not monopolise the time of websites. Indeed, a web crawler can aim to minimise its effect on websites. Second, web crawlers have to work with a pool of knowledge that includes several items of differing quality, including very low-quality objects generated to draw the web crawler and trick rating schemes.

One of the key issues is that a Web crawler only operates with partial knowledge, since it must assume the properties of the unknown sites depending on the portion of the Web already accessed. We interpret the problem of Web crawling as a method of finding related objects. In this case, as much information as possible about web pages is needed to be obtained by the web crawler.

Although the paradigm means that all aspects of the search engine can know all the web page properties, the architecture presented in this chapter is an attempt to divide these properties into smaller units for improved scalability (text, link graph, etc.). In the WIRE crawler, this architecture is introduced and the implementation of the WIRE crawler is detailed.

Benchmarking this architecture includes a mechanism that makes contrasts between various network, processor, memory and disc configurations, and we did not have any such benchmarks during this thesis. However, the results discussed in the following chapters regarding scheduling techniques, stop requirements and web characteristics are largely independent of the architecture selected.

There are several methods of separating a Web crawler's activities that may utilise the same crawling paradigm as outlined in sections 3 and 4 of this paper; the modularization described here shows that it is feasible to incorporate the model, and each task is easy enough to programme and debug the whole framework during the course of this study, although there are definitely other alternative architectures.

It is difficult to download all the Web pages in the form of today's Web, moreover, the number of Web pages is infinite, so the fraction of the Web that a crawler downloads should be the most relevant pages.

References

[1] Anish Gupta, K B Singh, A review on Effective Web crawling, International Journal of Advanced and Innovative Research, IJAIR, ISSN 2278-7844, 2013.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The Pagerank citation algorithm: bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[3] Anish Gupta, K.B.Singh, R. K. Singh. Study of Web Crawling Policies, International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2013.

[4] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *Proceedings of the eleventh international conference on World Wide Web*, pages 124–135, Honolulu, Hawaii, USA, May 2002. ACM Press.

[5] Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the Tenth Conference on World Wide Web*, pages 114–118, Hong Kong, May 2001. Elsevier Science.

[6] Anish Gupta, Priya Anand. Focused Web crawling and Its approaches, International Conference on Futuristic Trend on Computational Analysis and Knowledge Management, 2015, IEEE Digital Library.

[7] B. Liu and E. A. Fox. Web traffic latency: Characteristics and implications. *J.UCS: Journal of Universal Computer Science*, 4(9):763–778, 1998.

[8] Jerome Talim, Zhen Liu, Philippe Nain, and Edward G. Coffman Jr. Controlling the robots of web search engines. In *Proceedings of ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, pages 236–244, Cambridge, Massachusetts, USA, June 2001.

[9] Markus Sobek. Google dance – the index update of the Google search engine. http://dance.efactory.de/, 2003.

[10] Michelangelo Diligenti, Marco Maggini, Filippo Maria Pucci, and Franco Scarselli. Design of a crawler with bounded bandwidth. In *Alternate track papers & posters of the 13th international conference on World Wide Web*, pages 292–293. ACM Press, 2004.

[11] Dushyant Kumar, Dr. P.K. Dwivedi. A Study on In-Time-Frequency Algorithm. Cosmos: An International Journal of Management, 7(2): 1-3, 2018.

[12] Kumar, Puneet, "A Global Change in Education through Information Technology & Communication". Gyanodaya : The Journal of Progressive Education', pp 22-26, 2008.

[13] Khasim, Sayed, "The Discussion on Breaching Information Security", Cosmos Journal of Engineering & Technology, 4 (2): 29-33, 2014.

[14] Kumar, Puneet and Gupta, Ruchika, Information System's Security by using Matrices and Graphs, Conference proceedings on Information Security and Mobile Computing, pp. 62-66, 2008.

[15] Agarwal, Nidhi and Pundir, Neelam, "Information and Communication and Its Importance". Ambikeya Journal of Education, 8(1): 40-42, 2017.

[16] Dr. Sangeet Vashishtha, Pooja Sharma, Big Data- New Trend of Change in Complex Corporate World. Globus An International Journal of Management & IT, 10(1): 4-6, 2018.

[17] Agarwal, Nidhi and Gupta, Ruchika, "Role of Technology for the Efficiency of HR Management", Information and Communication Technology: Challenges & Business Opportunities, Excel India Publishers, Delhi, pp. 174-176, ISBN: 93-80697-95-3, 2011.

[18] Anuradha. "Study in Technological Challenges in Digital Libraries", Cosmos An International Journal of Art & Higher Education, 4(2): 9-11, 2015.

[19] Mishra, Shivani and Soni, Dr. Anita. "A Study on Technology, Thinking Styles, and Content of Education", Cosmos An International Journal of Art & Higher Education, 5 (2): 1-4, 2016.

[20] Adarsh Tiwari, Dr. Sudesh Kumar. A Study on Business Drivers to Receive Cloud Computing. Cosmos Journal of Engineering & Technology, 8(1), 1-3, 2018.

[21] Gupta, Ruchika and Kumar, Puneet. "Information Technology Business Value Assessment: A Case of State Bank of India". Globus: An International Journal of Management & IT, 42: 30-34, ISSN:0975- 721X, 2013.

[22] Sharma, Dr. Seema. "Technology, E-Learning and Social Media with Reference to Academic Achievement", Cosmos An International Journal of Art & Higher Education, 6 (1) : 7-8, 2017.

[23] Misra, Lisha and Saxena, Dr Aakash. A Study on Security Goals for Wireless Network. Cosmos Journal of Engineering & Technology, 8(1), 2018.
[24] Umesh A. Patel, Nita Brahmbhatt. "Implication of Six Sigma to Improve Library Services in Academic Libraries", Globus Journal of Progressive Education,4(2): 1-3, 2014.
[25] Kiruthiga Devi M, Dr. Sudesh Kumar. Artificial Intelligence, Machine Learning And Cognitive Computing with Profound Learning Technique. Globus An International Journal of Management & IT, 10 (1): 23 – 26, 2018.
[26] Dr Gandhi Singh Chauhan. Criteria To Select Library Automation Software. Cosmos: An International Journal of Management, 5(2): 1-5, 2016.
[27] Surishma Singh, Adnan Nasir. Digitalization of Education through Understanding of ICT. Globus Journal of Progressive Education, 8(1), ISSN: 2231-1335, 2018.

