Indian Journal of Science and Technology, *Vol 9(S1), DOI: 10.17485/ijst/2016/v9iS1/102825, December 2016*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improved Parallel Computation of PageRank for Web Searching

Hema Dubey[1], Nilay Khare[2] and Appu Kuttan KK[3]

[1,2]Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Link Road Number 3, Near Kali Mata Mandir, Bhopal - 462003, Madhya Pradesh, India; hema32150@gmail.com, nilay.khare@rediffmail.com
[3]Department of Mechanical Engineering, National Institute of Technology, NH 66, Srinivas Nagar, Surathkal, Mangaluru - 575025, Karnataka, India; pp_kuttan@Yahoo.com

Abstract

Background/Objectives: PageRank given by Brin and Larry in 1998, emerged as a dominant link analysis method used by web search engines for ranking of its search results. Efficient and fast computation of PageRank values for prodigious web graphs is indeed an important issue for web search engines today. Recognizing and fighting with spam web pages is also considered to be another necessary issue in web searching. **Methods/Statistical Analysis**: In this paper, we have proposed an efficient and accelerated parallel computation of PageRank scores on Graphics Processing Units (GPUs) which uses non even distribution of PageRank values. This work is experimented on datasets taken from Stanford Large Network Dataset Collection, on a system equipped with NVIDIA Quadro 2000 Graphics card using CUDA programming language. **Findings:** The proposed work has a speed up of 3.22 to 7.5 and is also capable of dealing with spam web pages. **Application:** The proposed algorithm helps in detecting spam web pages.

Keywords: CUDA, GPU, Parallel PageRank, Spam Web Pages

1. Introduction

There is bountiful information available on World Wide Web (WWW). Every day huge amount of data is added to WWW, which leads to change in web content and hyperlink structure of web. Web is therefore growing at a very fast pace. Due to the dynamic nature of WWW and bulk amount of information available on it, it is really a challenging task for web search engines to retrieve the relevant information from the web. The World Wide Web can be represented as a graph which contains huge collection of web pages and links between those web pages. The web pages are referred to as nodes and links between web pages are referred to as edges. When a query is fired by a user on the interface of search engine, he receives lot of web pages in response, of which some pages are more relevant in comparison to other pages. This gives rise to lots of ordering (or ranking) of web pages. PageRank was proposed by[1,2], subsequently Google company started using PageRank algorithm in its popular search engine. PageRank is recursive algorithm that verifies the importance of web pages by assigning a numeral weight to web pages. This algorithm determines a static value for each web page which implies that PageRank scores for web pages are calculated off-line and not at the query run time[3]. With extreme popularity of Google, the supremacy of PageRank has been proven over other hyperlink based techniques[4]. The PageRank scores must always be up-to-date despite of changing behaviour of World Wide Web. PageRank calculation must be performed in a shorter span of time as possible[5]. This gives rise to the requisite of performing PageRank computation using high performance computing tools. Many researches have been performed to solicit minimal possible time to compute

**Author for correspondence*

the PageRank scores of web pages. The advent of Graphics Processing Units (GPUs) pioneered a novel phase in the field of scientific computing. GPUs provide new alternatives to solve computationally intensive algorithms in fast and efficient way. It provides very valuable resources to parallelize and achieve optimum performance.

Thus PageRank computation can be accelerated with the use of parallelism and high memory bandwidth of GPU processing. In this paper, we have proposed an improved PageRank algorithm based on GPU framework for fast and efficient computation of PageRank scores. There is a speed up of about 3.22 to 7.5 in GPU based PageRank as compared to sequential CPU-based PageRank program.

The rest of the paper has been organized as follows. Section 2 makes a review on related work. Section 3 describes the background about multicore processors, PageRank algorithm and web graph representation. Section 4 discusses proposed parallel PageRank. The implementation details and experimental results are reported in Section 5. Conclusion is given in Section 6.

2. Literature Review

In the past years, many researches have been done on how PageRank can be improved[6], optimised[7] and personalised[8]. Although improvements in CPU based implementation of PageRank have been researched in detail in[9,10,11,12,13,14]. Researchers got attention in distributing PageRank computation over several computers recently. Research work on Parallel PageRank computation focuses on minimizing communication overhead and achieving faster convergence rate by exploiting the power of large cluster based computation. In[15] proposed a distributed PageRank calculation using Iterative Aggregation-Disaggregation (IAD) method through divide-and-conquer technique to effectively speedup the PageRank computation in a distributed environment. In[16] expedites the power method for PageRank computation on AMD GPUs by achieving 15x speedup on a Radeon 5870 Graphic Card compared with a PhenomII 965 CPU at 3.4GHz. In[17] used NVIDIA's SPMV libraries as CUBLAS; CUSPARSE for computation of PageRank on a Fermi-based GPU GTX480 and gain speed up of 3-4 times in terms of computational time. In[18] modelled the PageRank computation as a linear system and evaluated the performance of a parallel implementation on Beowulf cluster of 70 nodes. In[19] evaluated scalability of PageRank on the large-scale supercomputer TSUBAME which includes 96 Tesla GPUs, the best speed up on 64 GPUs achieved about 8 times faster than the CPU cluster implementation with the same number of CPU cores for 39.5 million web pages. In[20] implemented parallel PageRank computational architecture on a cluster of Opteron PCs networked via a Gigabit Ethernet. In[21] computed the approximate personal PageRank vectors in parallel using Pregel to illustrate the speed up over the serial method. In[22] made use of the fully distributed peer to peer architecture to speed up the PageRank computation. In[23] used partitioning and repartitioning techniques for computation of PageRank in parallel. In[24] proposed parallel computation of PageRank algorithm by using non uniform distribution of PageRank scores. In this paper, we have proposed a modified PageRank algorithm, and implemented it in both sequential and parallel manner to compare their execution time. We also state that the proposed parallel algorithm is immune to spam as compared to existing parallel PageRank algorithm.

3. Background

3.1 Multicore Processors

Multicore processors architectures are fabricated in order to enhance performance, curtail heat output and support more efficient simultaneous processing of multiple tasks by integrating two or more processor cores into a single processor socket. In this paper we describe GPU multi-core architecture which uses SIMD (Single Instruction Multiple Data) model for data parallelism. A GPU is a heterogeneous chip multi-processor (highly tune for graphics). General purpose computing on the GPU (Graphics Processing Unit) is an appealing area of research. As shown in Figure 1, GPU contains hundreds of cores that work in parallel for high performance computing. CUDA (Compute Unified Device Architecture) invented by NVIDIA is a heterogeneous scalable serial- parallel programming model and software environment for parallel computing. CUDA enables remarkable increase in computing performance by harnessing the power of the graphics processing unit (GPU). Thus, CUDA has made it possible to exploit the highly parallel and multithreaded environment of GPU's for parallel implementation of mathematical algorithms such as PageRank.

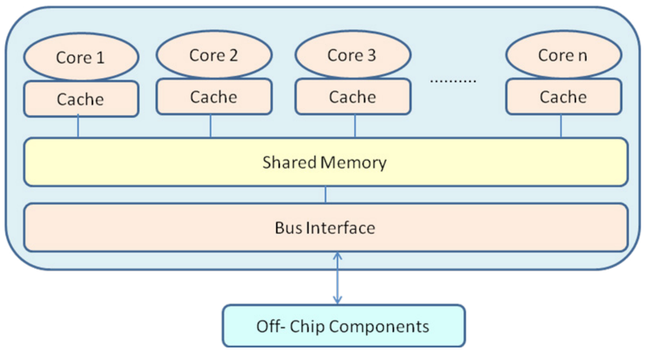

Figure 1. Multicore Processor Architecture.

3.2 Review of PageRank Algorithm

PageRank performs computing on a huge amount of data. The basic concept of PageRank technique is to disseminate significance from one page to another, through its outgoing hyperlinks. PageRank is mathematical model proposed by Google and is named on Larry Page, who is co-founder and president of Google. PageRank is a mathematical score that characterizes how significant a web page is on the internet. Google concludes that when a web page associates with other web pages, it means that it is actually voting for the other web pages. If large number of votes are cast for a web page, then that web page is said to be more important on world wide web. One thing is important that PageRank algorithm does not permit ranking of websites as a whole, whereas it determines the rank of each page separately. PageRank[1] represents probability distribution that corresponds to likelihood of a user clicking at random on hyperlinks to reach at any specific page. PageRank is an iterative algorithm used to determine the rank of a web page from the previous iteration's PageRank values. The original formula to calculate PageRank of a web page is given by:

$$\mathbf{r}_{i+1}(\mathbf{u}) = \sum_{v \in \mathrm{Su}} \frac{\mathbf{r}_i(v)}{Nv} \tag{1}$$

In which, $r_{i+1}(u)$ is the PageRank of page u at iteration i+1, Su is the set of pages pointing to u and Nv is the number of out links (recommendations) from page v. And $\mathbf{r}_i(v)$ is the PageRank of page v at iteration i.

Initially PageRank values of all pages are assigned to 1. Since PageRank is an iterative method, PageRank calculation is repeated until PageRank scores converge to the final stable values. Nevertheless, there arises a problem with such definition of PageRank. Practically, some web pages will accumulate more and more PageRank scores after each iteration and repudiate to share to other web pages. Because of this there will be PageRank value sinking to zero at the end of iterative process. This problem occurs due to the web pages (nodes) that do not have any out links in the graph. Thus, the nodes with zero outdegree are referred to as dangling nodes (also known as hanging pages, zero-out-link and web frontier).

Larry Page and Sergey Brin introduce some modifications to deal with dangling nodes: firstly, dangling nodes are replaced by nodes linking to all other nodes; and secondly, damping factor is added which represents random walks of the random surfer process of the graph. This results into following modification of Equation (1):

$$\mathbf{r}_{i+1}(\mathbf{u}) = (1 - \alpha) + \alpha \sum_{v \in \mathrm{Su}} \frac{\mathbf{r}_i(v)}{Nv} \tag{2}$$

Where α is a damping parameter used to represent the probability of a surfer trailing hyperlinks of the web graph, its value is set between 0 and 1. In this improved version of PageRank model, the random surfer has two options:

(a) With probability α, he randomly chooses an out link to follow. (b) With 1– α probability, he skips to a random web page without a link.

3.3 Web Graph Representation

The input dataset for PageRank calculation is a web graph. The web graph consists of web pages and hyperlinks. The web page is referred to as nodes and hyperlinks are referred to as edges. The web graph is huge in size containing millions to billions of nodes (web pages) and the number of edges (hyperlinks) is about ten times more. It is therefore important to choose an efficient data structure for the representation of web graph. Computing the PageRank score for a large web graph using in memory matrix format is certainly not feasible. A common solution is to use "link structure file" storage technique that

Table 1. Link Structure File

Source id	Out degree	Destination id
1	3	22; 34; 95
2	4	1000; 454; 200; 3
3	1	32
4	0	
5	2	28889; 36777
…..	…..	…..

stores the links in a format like "Source Page id, out degree, Destination Page id".

Table 1 illustrates an instance of a "link structure file" which represents a web graph. In this table each row represents a record in the file. The first record contains information of the web page with id 1 that has out-degree of 3, which means that this web page points to 3 other web pages. Similarly the other records contain information about other web pages.

4. Improved Parallel PageRank Algorithm

In PageRank Algorithm, one web page's PageRank value is evenly divided among its entire outgoing links. Thus this algorithm works on uniform distribution of PageRank values. Due to this factor, sometimes the PageRank scores of less important web pages (spams) exceeds and PageRank scores of important web pages degrades. In this paper we present an improved parallel PageRank model which uses non uniform distribution of PageRank values to deal with spam web pages. Assuming there are *p* pages (*Y*1, *Y*2,..... *Yp*) pointing to *u*. Furthermore, *Xij* are the web pages out linked from *Yi* (its number is ||*Yi*||).

$$PR(u) = \frac{1-d}{N} + d \times \sum_{i=1}^{p} \frac{PR(u)}{\sum_{j=1}^{\|Y_i\|} PR(X_{ij})} \times PR(Y_i) \qquad (3)$$

Here, PR(*u*) is the PageRank value of web page *u*, PR(Y_i) is the PageRank value of web pages Y_i pointing to u and PR(X_{ij}) is PageRank value of page X_{ij} which is outgoing link of Y_i.

$$\frac{PR(u)}{\sum_{j=1}^{\|Y_i\|} PR(X_{ij})} \text{ is the part page } Y_i \text{ contributes to } u$$

This algorithm intensifies PageRank values of important pages and brings down PageRank values of less important pages which are usually spam nodes. The sequential implementation of equation (3) without using any standard datasets is presented in[25]. We have implemented equation (3) in sequential manner algorithm by using standard datasets from Stanford Large Network Dataset Collection[26]. We have also implemented the parallel version of this algorithm using the same standard datasets so as to improve the PageRank computation of web pages. Then we have finally compared the execution time of both sequential and parallel version of proposed algorithm. The speed of computing PageRank scores on GPU is 3.22 to 7.5 times faster than that of sequential implementation. In this paper, we have also compared the PageRank scores of Parallel PageRank and Proposed Parallel PageRank, the results show that PageRank values of relevant web pages increases and PageRank values of irrelevant web pages (spam web pages) decreases. This helps in showing important web pages on the top list of search engine.

Pseudocode for Proposed Parallel PageRank is:

- Input: Let G represents set of web pages or nodes
- Output: A file showing PageRank for each web page
- for each node n in Graph do in parallel
- Initially set n.prev_PR := 1.0;
- function PR(G)
- for 1 to k do
- for each node n Graph do in parallel
- n.PR := 0;
- for each page p in n.inlinkneighbors do
- sum:=0.0
- for each page q in p.outlinkneighbors do
- sum += q.prev_PR)
- *n*.PR += (1-d) + d * (p.PR)
- difference := 0
- for each node n Graph do in parallel
- difference := maximum (n.PR - n.prev_PR, mx)
- if difference < threshold do
- stop the program
- for each node n Graph do in parallel
- n.prev_PR := n.PR

5. Implementation Details

We have implemented the Sequential Proposed PageRank (equation 3) in C++ programming language. Parallel implementation of PageRank (equation 2) is done in CUDA programming model and Proposed Parallel PageRank (equation 3) is also implemented in CUDA programming language. All the three implementations are tested on standard datasets taken from Stanford Dataset collection[25]. The execution time of sequential and parallel PageRank is compared, which shows that PageRank computation is accelerated by using GPU CUDA. Then we compare Parallel PageRank and Proposed Parallel PageRank to verify that proposed algorithm boost the PageRank scores of relevant web pages and reduce the

PageRank scores of irrelevant web pages (which are usually spam pages). The experimental results prove that proposed algorithm is effective and immune to spam. Figure 2 depicts how PageRank computation is performed using GPU CUDA. GPUs provide better suppleness, additional registers, compatibility with lengthy programs, and supports control-flow primitives for example loops, branches and subroutines. In GPU, calculation of PageRank consists of mainly two parts, host and kernel. Execution starts with the Host program, Host program reads the input web graph and copies it to the GPU's memory. Host invokes the kernel to be executed on GPU for computation of partial PageRanks with N source web pages. It is noted that kernels calls are non-blocking for Host, therefore kernel and Host can run simultaneously. When GPU starts processing, concurrently Host also starts computing PageRanks of web pages having in-degree more than N. When computation of ranks on both GPU and Host are completed, immediately partial PageRanks of all web pages from Host are brought into the GPU memory and added with partial PageRanks computed at GPU memory. In this manner, new PageRank values of all web pages are computed and input vector is updated for next iteration both at Host and GPU memory.

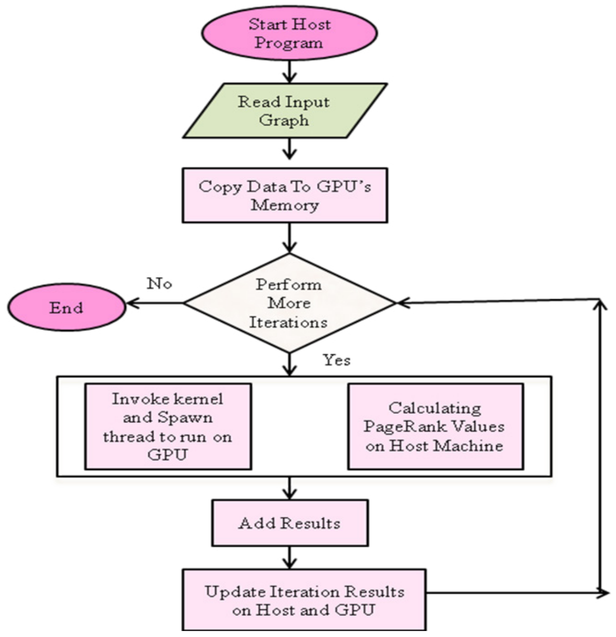
Start Host Program
Read Input Graph
Copy Data To GPU's Memory
Perform More Iterations
No
End
Yes
Invoke kernel and Spawn thread to run on GPU
Calculating PageRank Values on Host Machine
Add Results
Update Iteration Results on Host and GPU

Figure 2. PageRank Computation in GPU CUDA.

5.1 Experimental Environment

We have been implemented and tested the serial and parallel PageRank Algorithms on a system equipped with:

- Processor: Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz
- Host Memory: 4 GB
- CPU cores: 12
- 2 dual graphics cards: NVIDIA Quadro 2000 and NVIDIA Tesla C2075
- Multiprocessors: 14

5.2 Input Dataset

We have performed experiments on four datasets taken from Stanford Large Network Dataset Collection[25]. The first dataset Email-EuAll contains 265,214 web pages and 420,045 hyperlinks. The second dataset Amazon0302 contains 262,111 pages and about 1 million hyperlinks. The third dataset Amazon0312 includes about 400,727 web pages and 3 million hyperlinks. The fourth dataset Wiki-Talk is even larger with about 2 million pages and 5 million hyperlinks. We summarize statistic of our datasets in Table 2.

6. Experimental Results

6.1 Speed Up in Execution Time

We performed experiments on CPU and GPU with four datasets mentioned in Table 2. Specifically, we measure their execution (running) time. The parallel algorithms implemented on GPU using CUDA programming language show that the speed of computing PageRank scores increases markedly when compared to CPU-based version. Table 3 compares the running time in seconds of PageRank computation on GPUs with that on the CPU.

As we can see from the Table 3, parallel PageRank gains a speed up of about 3.2 to 7.5 times than that of sequential CPU-based program for four different standard datasets.

6.2 Spam Web Pages Detection

The proposed parallel PageRank Method is immune to spam web pages. The proposed scheme uses non uniform distribution of PageRank values to enhance the PageRank scores of important web pages and lessen the PageRank scores of spam web pages (irrelevant web pages). We have implemented existing PageRank algorithm and proposed PageRank algorithm in GPU CUDA, and then compare the PageRanks of both algorithms.

We have shown the experimental results in 4 graphs given below (Figures 3, 4, 5, 6). In Figure 3, we have used

Table 2. Statistic of four experimental datasets[25]

Datasets		Type	Nodes	Links	Web Graph Size (MB)	Description
1	Email-EuAll	Directed Graph	265,214	420,045	4.76	Email network from a EU research institution
2	Amazon0302	Directed Graph	262,111	1,234,877	16.4	Amazon product co-purchasing network from March 2 2003
3	Amazon0312	Directed Graph	400,727	3,200,440	43.3	Amazon product co-purchasing network from March 12 2003
4	Wiki-Talk	Directed Graph	2,394,385	5,021,410	63.3	Wikipedia talk communication network

Table 3. Execution time (in seconds) of experiments performed on the four datasets

Datasets		Sequential PageRank Equation (3)	Parallel PageRank Equation (2)	Proposed Parallel PageRank Equation (3)	Speed Up
	Running Time (in Seconds)				
1	Email-EuAll	45	6	6	7.5
2	Amazon0302	35	5	5	7
3	Amazon0312	80	13	13	6.15
4	Wiki-Talk	145	45	45	3.22

the standard dataset Email-EuAll, which includes 265,214 nodes and 420,045 links. In Figure 4, Amazon0302 standard dataset is used which contains 262,111 nodes and 1,234,877 links. Figure 5 shows implementation of standard dataset Amazon0312 that has 400,727 nodes and 3,200,440 edges. And finally Figure 6 uses Wiki-Talk dataset. We have shown PageRank values of only few nodes in all the 4 Figures viz. 3, 4, 5, and 6 for the ease of visualization. All the 4 Figures (graphs) depicts that PageRanks of important web pages are enhanced and PageRanks of spam web pages are reduced by the proposed Parallel PageRank Algorithm.

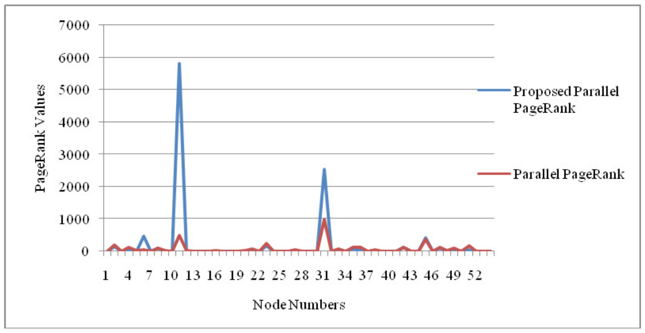

Figure 3. PageRank values of some nodes of dataset 1: Web graph – "Email-EuAll".

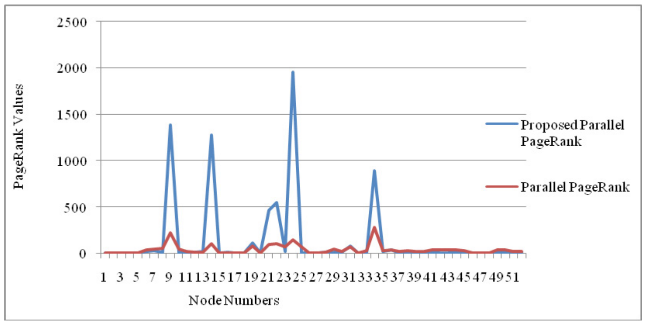

Figure 4. PageRank values of some nodes of dataset 2: Web graph – "Amazon0302".

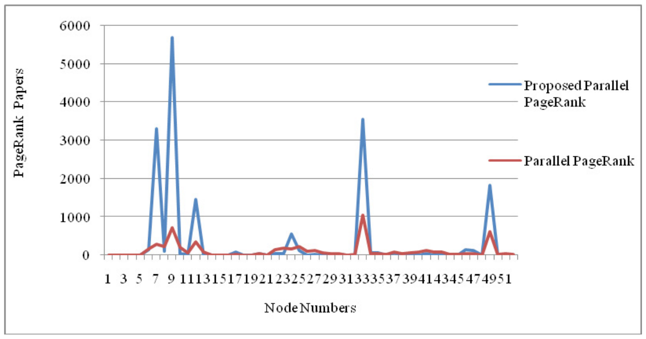

Figure 5. PageRank values of some nodes of dataset 3: Web graph – "Amazon0312".

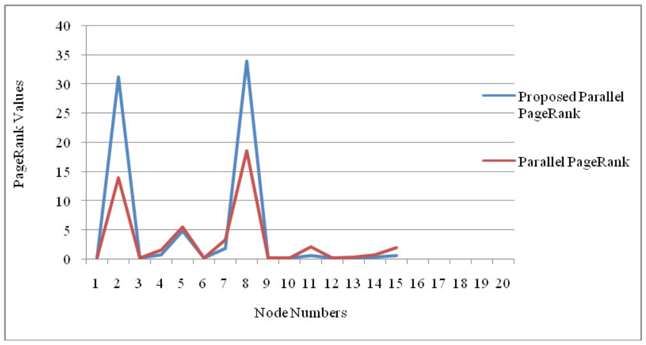

Figure 6: PageRank values of some nodes of dataset 4: Web graph – "Wiki-Talk".

7. Conclusion

In this paper, we have proposed a parallel implementation of not-equal-distribution of PageRank scores using CUDA Programming language in order to accelerate the PageRank computation and reduced the PageRanks of spam web pages, so that users can get more accurate web pages when firing a query in the Google search engine. We performed experiments on CPU and GPU with various standard datasets to examine the speed up capability. The experimental outcomes are pretty fascinating and show that speed of computing PageRank scores by the proposed approach gives a considerably greater performance than CPU based algorithm. The proposed scheme is also immune to spam pages.

8. Acknowledgement

This research was conducted in High Performance Computing Laboratory of Computer Science Department of Maulana Azad National Institute of Technology, Bhopal.

9. References

1. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Technical report. Stanford Digital Library Technologies Project. 1999.
2. Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Proceedings 7th WWW Conference Elsevier. 1998; 30(1-7):107-17.
3. Liu B. Exploring hyperlinks, contents and usage data. Berlin: Springer-Verlag. Handbook of Web Data Mining. 2011; p. 247-77.
4. Duong NT, Nguyen QAP, Nguyen AT, Nguyen HD. Parallel PageRank computation using GPUs. New York, USA: Proceedings of the Third Symposium on Information and Communication Technology ACM. 2012 Aug; p. 223-30.
5. Gleich DF. PageRank Beyond The Web. SIAM Review. 2015; 57:321-63.
6. Abiteboul S, Preda M, Cobena G. Budapest, Hungary: Proceedings of the 12th international conference on World Wide Web ACM. Adaptive On-Line Page Importance Computation. 2003 May; p. 280-90.
7. Liu D, Gong Y. Optimal methods of PageRank Algorithm on the bilingual web page. Chengdu, China: Proceedings of the 2nd International Conference on Computer Engineering and Technology. 2010 April; 689-91.
8. Haveliwala TH. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Transactions on Knowledge and Data Engineering. 2003 July; 15(4):784-96.
9. Dubey H, Roy BN. An Improved Page Rank Algorithm based on Optimized Normalization Technique. International Journal of Computer Science and Information Technologies. 2011 Sep; 2(5):2183-88.
10. Singh AK, Kumar R, Leng AGK. Efficient Algorithm for Handling Dangling Pages using Hypothetical node. Seoul, South Korea: IEEE 6th International Conference on Digital Content, Multimedia Technology and its Applications. 2010 Aug; p. 44-9.
11. Xing W, Ghorbani A. Weighted PageRank Algorithm. Proceedings of IEEE 2nd Annual Conference on Communication Networks and Services Research. 2004 May; p. 305-14.
12. Al-Saffar S, Heileman G. Experimental bounds on the usefulness of personalized and topic-sensitive pagerank. Fremont, CA: International Conference on Web Intelligence. 2007 Nov; p. 671-75.
13. Rungsawang A, Puntumapon K, Manaskasemsak B. Un-biasing the link farm effect in pagerank computation. Niagara Falls, ON: IEEE 21th International Conference on Advanced Networking and Applications. 2007 May; 924-31.
14. Yuan F, Yin C, Liu J. Improvement of PageRank for Focused Crawler. Qingdao: IEEE Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. 2007 July; 2(7):797-802.
15. Zhu Y, Ye S, Li X. Distributed pagerank computation based on iterative aggregation-disaggregation methods. ACM, Bremen, Germany: Proceedings of the 14th ACM international conference on Information and knowledge management. 2005 Oct; 578-85.
16. Wu T, Wang B, Shan Y, Yan F, Wang Y, Xu N. Efficient PageRank and SpMV Computation on AMD GPUs. San Diego, CA: IEEE 39th International Conference on Parallel Processing. 2010 Sep; p. 81-9.

17. Praveen K, Vamshi Krishna K, Anil Sri Harsha B, Balasubramanian S, Baruah PK. Cost Efficient PageRank Computation using GPU. Bengaluru, India: International Conference on High Performance Computing (HiPC). Student Research Symposium. 2011 Dec.
18. Gleich D, Zhukov L, Berkhin P. Fast parallel PageRank: A linear system approach. Purdue Universit: Technical Report. 2004.
19. Cevahir A. Aykanat C. Turk A, Cambazoglu BB, Nukada A, Matsuoka S. Efficient PageRank on GPU clusters. IPSJ SIG Technical Report, HPC-128. 2010.
20. Manaskasemsak B, Rungsawang A. Parallel PageRank Computation on gigabit PC Cluster. Proceedings of the 18th IEEE International Conference on Advanced Information Networking and Applications AINA. 2004 March; 1(0):273–77.
21. Perozzi B, McCubbin C, Halbert JT. Scalable graph clustering with parallel approximate PageRank. Springer-Verlag Wien: Social Network Analysis and Mining. 2014 March.
22. Sankaralingam K, Sethummadhavan S, Browne JC. Distributed PageRank for P2P Systems. Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing. 2003 June; p. 58-68.
23. Cevahir A, Aykanat C, Turk A, Cambazoglu BB. Site-Based Partitioning and Repartitioning Techniques for Parallel PageRank Computation. IEEE Transactions on Parallel and Distributed Systems. 2011 May; 22(5):786-802.
24. Dubey H, Khare N, Appu Kuttan KK, Bhatia S. Improved Parallel PageRank Algorithm for Spam Filtering. Indian Journal of Science and Technology. 2016 Oct; 9(38).
25. Date accessed 03/02/2016: Available from: https://snap.stanford.edu/data/.
26. Bing-Yuan Pu, Ting-Zhu Huang, Chun Wen, An improved PageRank algorithm: immune to spam. Melbourne: Fourth International IEEE Conference on Network and System Security, VIC, no. 978-0-7695-4159-4. 2010 Sep; p. 425-29. DOI:10.1109/NSS.2010.12