Efficient Algorithm for Handling Dangling Pages Using Hypothetical Node

Ashutosh Kumar Singh
Department of Electrical and Computer Engineering
Curtin University of Technology
Miri, Malaysia
ashutosh.s@curtin.edu.my

Ravi Kumar P
Department of Electrical and Computer Engineering
Curtin University of Technology
Miri, Malaysia
ravi2266@gmail.com

Alex Goh Kwang Leng
Department of Electrical and Computer Engineering
Curtin University of Technology
Miri, Malaysia
alexgoh.kwangleng@gmail.com

***Abstract*—Dangling pages are one of the major drawbacks of page rank algorithm which is used by different search engines to calculate the page rank. The number of these pages increasing with web as the source of knowledge. However search engines are the main tools used in knowledge extraction or Information retrieval from the Web. We proposed an algorithm to handle these pages using a hypothetical node and comparing it with page rank algorithm.**

Keywords–PageRank; Dangling Pages; zero-one-gap; Penalty Pages; Link rot.

I. INTRODUCTION

The PageRank algorithm of Page et al. [1] has taken a new step in ranking of the results produced by search engines. This PageRank algorithm is developed by Brin and Page of Stanford University during their Ph D using citation analysis [11, 12]. The PageRank algorithm is a link-based ranking method and it is used in the popular Google search engine. The ranking order is purely determined by the link structure of the Web. Google search engine work in this way, imagine a random surfer surfing the Web, going from one page to another page by randomly choosing an outgoing link from one page to go into the next page. This can lead to dead end at pages with no outgoing links. So, a certain fraction of the time, the surfer can simply choose a random page from anywhere on the Web. This theoretical random walk of the Web is a *Markov Chain* [10] or *Markov process*. The limiting probability that a dedicated random surfer visits any particular page is its PageRank. A page has high PageRank if it has links to and from other pages with high rank. Even though PageRank algorithm is so popular, there are also some problems associated with the PageRank algorithm. One such problem is the "dangling page" i.e. web pages not having any outgoing links or pages without any hyperlinks in the web. It can also be called as hanging pages. Dangling pages can be categorized into the following four categories based on how the dangling pages occurs, pages that are protected by robots.txt, pages not having genuine outlink like the pdf files, images etc., URLs with meta tag indicating that links should not be followed from the page, and pages that return a 500 class response at crawl time due to configuration problems, network problems and bad link problems. According to N. Eiron et al. [2], the number of dangling pages in the Web is keep growing. So dangling pages can not be omitted, they may contain quality information and one of the objective of knowledge extraction is not to miss out any information from any sources. The presence of the dangling pages can cause philosophical, storage and computational problems for PageRank. The problems are discussed later in the experimental section. For example let us consider the following snap shot of a Web as shown in Figure 1.

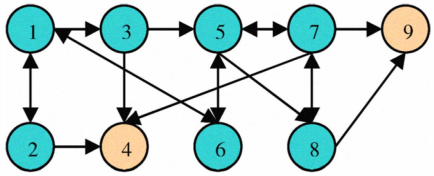

Figure 1 A snap shot of a web

In the above snap shot of a Web graph, nodes 4 and 9 are dangling nodes because there is no outgoing links from node 4 and 9. All the other nodes are non-dangling nodes. In this paper nodes and pages are used interchangeably and it means the same.

Dangling page problem is discussed by many researchers Page et al. [1], N. Eiron et al. [2], Amy N. Langville et al. [3], M. Bianchini et al. [4] and Xuanhui Wang et al. [5].

This paper is organized as follows. Next Section shows previous work on the dangling pages and it also includes some recent works on the dangling pages. Section III provides our proposed method on handling dangling pages and the matrix model of it. Section IV provides our experimental results of the proposed method. This paper is concluded in Section V.

II. PREVIOUS WORK

A. Dangling Pages

Dangling pages are those Web pages they don't have any outgoing links or if the outgoing links are unknown but have incoming links i.e. the pages are being referenced by other pages and they are not referring to other pages. It is also referred as hanging pages, zero-out-link [5] and web frontier [2]. Such hanging pages can act as sinks or black holes for PageRank, a key factor in the Google search algorithm. When the original PageRank algorithm was developed, the Web is mostly a static one using human edited documents residing in

the file systems. So it is easy to crawl and index the entire Web. There are not many changes in the link structure of the Web. Recently Web is evolving into a dynamic one driven by databases and the Web is growing in a big way. Also the link structures are being altered due to the Web maintenance and other activities in the Web. Link rot [2] is a kind of problem associated with dangling pages. Here, the links have worked at one time are not working now. It is because the links are broken, the contents are removed or the URL is being changed. That makes the crawling is bit difficult for the search engines particularly the frontier of the Web. The number of dangling pages keeps increasing in the Web due to the above reasons and other reasons. Dangling nodes account for one-fourth of the Web's nodes [3].

B. Significant work on dangling pages

Here we discuss some of the previous methods discussed for handling dangling pages. In the original PageRank algorithm proposed by Brin and Page [1], the dangling pages are removed from the graph and the PageRank is calculated for the non-dangling pages. After calculating the PageRank, the dangling pages can be added back in without affecting the results. The authors claim that few iterations are enough to remove most of the dangling pages.

S. D. Kamvar et al. [6] suggested that by removing the dangling nodes and then re-inserting them for the last few iterations. Completely removing all the dangling pages will alter the results on the non-dangling pages to some extent since the out-degrees from the pages are adjusted to reflect the lack of links to dangling pages. This approach is supported by S. Brin et al. [7] and T. Haveliwala [8].

The process of removing dangling nodes may produce again new dangling nodes, and the process needs to be repeated iteratively until no dangling nodes remain. In our proposed method, we keep the dangling nodes and did the computation and produces the ranking of all the pages including the dangling pages to stick to our basic concept of not loosing any information from any sources.

S. D. Kamvar et al. [6] suggested that to jump to a randomly selected page with probability 1 from every dangling page. For example the nodes V of the graph ($n = |V|$) can be partitioned into two subsets:

- S corresponds to a strongly connected subgraph ($|S| = m$).
- The remaining nodes in subset D have links from S but no outlinks.

A virtual node (n+1) is added to which the random jumps may be made. The new node set is denoted by $V' = V \cup \{n + 1\}$. New edges $(i, n + 1)$ are added. In that $i \in D$ and $(n + 1, j)$ for $j \in S$ to define an expanded edge set ε'. PageRank of the nodes in V' can be computed via the principal eigenvector by partitioning the matrix and vector.

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \alpha S & O & e/m \\ \alpha D & O & 0 \\ (1-\alpha)e^T & e^T & 0 \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \qquad (1)$$

where, if d_j is the out degree of node j

$$s_{ij} = \begin{cases} d_j^{-1} & \text{if } (i, j) \in \varepsilon \text{ and } i, j \in S \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ij} = \begin{cases} d_j^{-1} & \text{if } (i, j) \in \varepsilon \text{ and } i \in S, j \in D \\ 0 & \text{otherwise} \end{cases}$$

and p, q, r are of the row dimension of S and D and 1 and e is the vector of 1's of conforming dimension. The individual equations are:

$$p = \alpha S p + (r/m)e \qquad (2)$$

$$q = \alpha D p \qquad (3)$$

$$r = (1-\alpha)p + e^T q \qquad (3)$$

$$= \{(1-\alpha)e^T + \alpha e^T D\}p \qquad (4)$$

This structure can be used to compute p and q from a reduced eigen-system.

$$\begin{pmatrix} \hat{p} \\ \hat{r} \end{pmatrix} = \begin{pmatrix} (1-\alpha)e^T + \alpha e^T D & e/m \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{p} \\ \hat{r} \end{pmatrix} \qquad (5)$$

Since the reduced matrix is column stochastic and

$$\hat{p} = \alpha S \hat{p} + \left(\hat{r}/m\right)e \qquad (6)$$

$$\hat{r} = \{(1-\alpha)e^T + \alpha \alpha^T D\}\hat{p} \qquad (7)$$

Now $\hat{p}$ and $\hat{r}$ can be solved in equation (7) by a standard iterative method like power iteration and then $\hat{q}$ can be calculated.

$$\hat{q} = \alpha D \hat{p} \qquad (8)$$

Link rot is a kind of problem associated with dangling pages. Here, the links have worked at one time are not working now. It is because the links are broken, the contents are removed or the URL is being changed. This will return with HTTP code 403 or 404. The pages that return this 403 or 404 HTTP code are called as *penalty pages* [2]. According to the authors, during the crawling process of over a billion pages it was found that approximately 6% of all web pages that were linked to turned out return a 404 code. This penalty pages may

have the effect on the ranks of nearby pages. There are several algorithms "push-back", "self-loop", "jump-weighting" and BHITS" proposed by N. Eiron et al. [2] based on the basic PageRank algorithm to adjust the ranks of pages with links to penalty pages.

The "push-back" algorithm works as follows, that if a page has a link to a penalty page, then it should have its rank reduced by a fraction, and the excess rank from this page should be returned to the pages that pushed rank to it in the previous iteration. The main purpose is that of limiting the "inflow" of rank to such pages.

In the "self-loop" algorithm, the authors add a self-loop link to itself, and with some probability γ_i follow this link. The probability γ_i should be smaller if the page has a large number of outlinks to penalty pages. In this way, a page that has no bad outlinks will retain some of its own rank by following a link to itself, whereas a page with only bad links will not retain any of its rank this way.

In "jump-weighting" approach, the penalty pages are treated as dangling nodes in the web graph, and collapsed them into the virtual node along with the legitimate dangling nodes, in equation (1). The standard procedure then redistributes the rank of the virtual node evenly (or to a chosen seed set). The N. Eiron et al. [2] propose an alternative procedure of biasing the redistribution so that penalized pages receive less of this rank. Using the above notation for good and bad pages, a straight-forward choice is to weight the link form the virtual node to an unpenalized node in C (or the seed set) by ρ and to a penalized node by $\rho g_i/(g_i + b_i)$, where ρ is chosen so that the sum of all these edges weights is unity.

The BHITS algorithm resembles the HITS algorithm [13], in the sense that it is derived from a random walk in both the forward and backward directions. In contrast to HITS algorithm, the BHITS algorithm does not depend on a query, and ranks all pages together. In this way it is more similar to the hub walk of SALSA [14]. As in the other algorithms, the purpose is that pages pointing to penalty pages should be degraded to some extent. The algorithm is described as a random walk that uses a forward step as in the normal PageRank, followed by a "backward" step for dangling nodes. For all non-dangling nodes, the backward step consists only of a self-loop. The authors distinguish two cases for the backward step from a dangling node. In the case of a penalty page, the authors forward all of its score to the virtual node. In the case of a non-penalty page, the backward step would divide the current score of the page by the number of in-links and propagate its score equally among all of the backward links. The purpose of this is to return the rank of pages that point to non-penalty pages, but to redistribute the rank that is given to penalty pages.

A fast two-stage algorithm for computing PageRank is proposed by Chris Pan-Chi Lee et al. [9] is based on the Markov chain reduction. The PageRank vector is considered as the limiting distribution of a homogeneous discrete-time Markov chain that transitions form one web page to another web page. The authors provided a fast algorithm for computing this vector. This algorithm uses the "lumpability" of Markov chain and constructed in two stages. In the first stage, the authors compute the limiting distribution of a chain where only the dangling pages are combined into one super node. In the second stage, the authors compute the limiting distribution of a chain where only the non-dangling pages combined. When this two limiting distributions are concatenated, the limiting distribution of the original chain, the PageRank vector is produced. According to the authors, this method can dramatically reduce the computing time and is conceptually elegant.

There is another problem associated with dangling pages is "zero-one-gap" problem proposed by Xuanhui Wang et al. [5]. This is one of the recent developments in this dangling page problem. The zero-one-gap problem refers to the number of out-links a page has. According to the authors there is a big difference between pages having 0 and 1 out link. This is because of the probability of jumping to random pages is 1 in a zero-out-link page and this drops to α for a page with a single out-link. This zero-one-gap problem allows the spammers to easily manipulate PageRank results by setting up bogus link pages to solely increase the PageRank results. Xuanhui Wang et al. [5] proposed a DirichletRank algorithm based on the Bayesian estimation of transition probabilities. The main objective of DirichletRank is to solve the zero-one-gap problem. According to the authors, apart from solving the zero-one-gap problem, it also solves the zero-out-link problem i.e. the dangling pages.

III. PROPOSED ALGORITHM

Our proposed method to handle the dangling pages is to connect a hypothetical node to the graph and make the hypothetical node to point back to itself. Then make all the dangling nodes to point to that hypothetical node. A similar approach is proposed by M. Bianchini et al. [4].

The basic PageRank model treats the whole Web as a directed graph $G(V, E)$, with a vertex set of V of N pages and a directed edge set E. The directed graph can be represented as matrix. PageRank creates graph and matrix before it computes the rank. Consider the following generalized matrix M.

$$M = \begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3}\cdots & X_{1,n} & X_{1,h} \\ X_{2,1} & X_{2,2} & X_{2,3}\cdots & X_{2,n} & X_{2,h} \\ X_{3,1} & X_{3,2} & X_{3,3}\cdots & X_{3,n} & X_{3,h} \\ X_{n,1} & X_{n,2} & X_{n,3}\cdots & X_{n,n} & X_{n,h} \\ X_{h,1} & X_{h,2} & X_{h,3}\cdots & X_{h,n} & X_{h,h} \end{bmatrix}$$

The above matrix is an $m \times m$ matrix where $m = (n+1)$ i.e. the last column and the last row is the hypothetical one which we will be used in dealing with the dangling pages. In our experiment, we used a sample directed graph with 6 nodes as shown in Figure 2.

In the directed graph (Figure 2), there are 6 nodes. Nodes X_1, X_2, X_3 and X_6 are non-dangling pages (shown in blue color). X_4 and X_5 are dangling nodes (shown in tan color) i.e. they do not have any outgoing links. Our research here is to show the effect of dangling nodes X_4 and X_5 in the PageRank computation. Also it shows how the neighboring pages rank gets affected because of the dangling pages. An adjacency

matrix is created for the directed graph as shown in Figure 2 which is using the simple adjacency matrix rule in form of row and column.

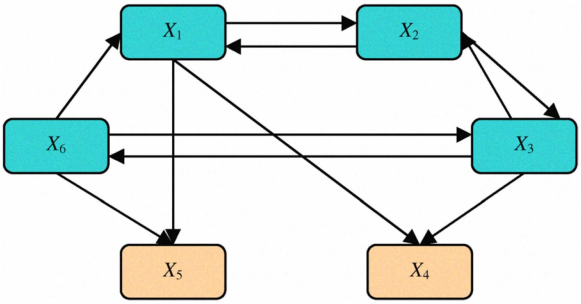

Figure 2 A Directed Graph showing 6 pages of web

This matrix also can be called as Link matrix in the Web. The adjacency matrix A of G (for the graph in Figure 2) is an $m \times m$ zero-one matrix with 1 as its $(i, j)^{th}$ entry when X_i and X_j are adjacent i.e. there is a connection between i and j and 0 as its $(i, j)^{th}$ entry when they are not adjacent i.e. there is no connection between i and j. In other words, the adjacency matrix, $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if there is a connection between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

In the above Matrix A, rows 4 and 5 are having only zeros. It means that node 4 and node 5 are dangling nodes. Matrix A is not stochastic and it needs to be stochastic as per the PageRank model. Using our proposed method, i.e. connect a hypothetical node with self loop and all the dangling nodes point to that hypothetical node (gray color) is shown in Figure 3, matrix A can be stochastic.

The adjacency matrix or the link matrix for the graph in Figure 3 is as follows by including the hypothetical node in the matrix:

$$\bar{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

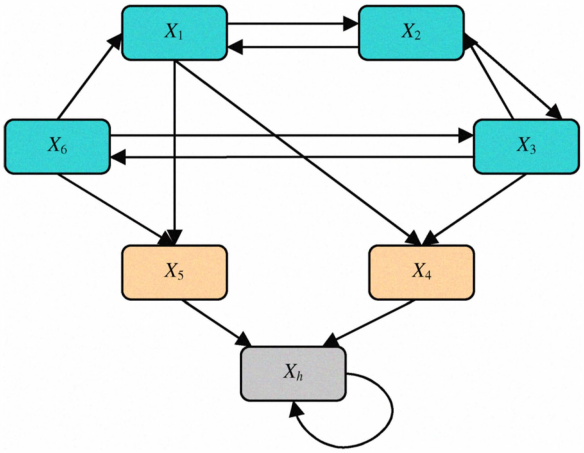

Figure 3 A Directed Graph including the hypothetical node X_h

This matrix $\bar{A}$ is stochastic now and it is a PageRank model. All the rules that can be applied to the PageRank model can also be applied to this matrix now. We consider two cases, Case 1 is using the original PageRank algorithm, where the dangling pages are excluded in the computation using the graph structure. Case 2 is using our proposed method to handle the dangling pages by including a hypothetical node with self loop and all the dangling nodes to point to that hypothetical node.

IV. Experimental Results

We used the sample directed graph with 6 nodes as the input to our PageRank program, implemented in Java. The program is tested on an Intel Core 2 (2.40 Ghz) with 4GB RAM. The program is a user friendly one and the number of nodes and the number of outgoing links for each node can be entered as an input. The program produces PageRank values for each node as the output after the convergence. The output is produced in the Excel CSV format. Because of the limited resources we could not apply the PageRank program on a real Web, instead we just used the sample Graph shown on Figure 2. The program pseudo code is given below:

Pseudo Code

```
Main Procedure
Initialize checkIteration is true
DO
    call PageRank to calculate the PageRank for every nodes
    save the PageRank for every nodes
    If the PageRanks of last Iteration has the same PageRanks with
    current Iteration
    checkIteration is false
WHILE quits when checkIteration is false
Procedure PageRank
    Initialize result to 0.15 (1 - the damping factor)
FOR every outgoing nodes of the current node
    call Calc
Add up result with the results from Calc of all outgoing nodes
Procedure Calc
    Calculate the result by getting the PageRank of the current node
    divide by the numbers of outgoing links of the current node times
    0.15 (1 - the damping factor)
```

Case 1

In the experiment case 1, we removed the dangling pages as it is done in the original PageRank program and ran the program. The output shows only the non-dangling nodes X_1, X_2, X_3 and X_6. The output of the experiment case 1 (PageRank iteration) is shown below as a chart in Figure 4.

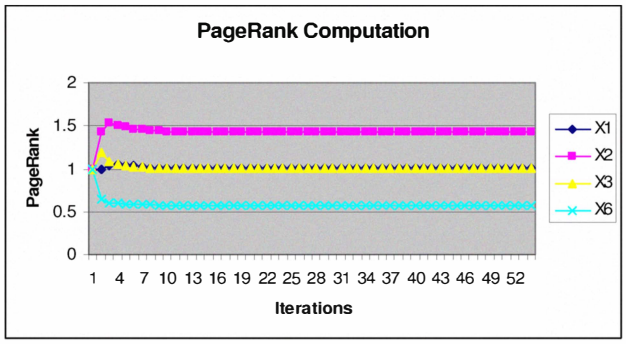

Figure 4 PageRank computation chart by excluding dangling nodes

In the figure 4, the dangling pages are completely removed and we lose the ranking of those dangling pages. But it is not accurate because the dangling nodes might be having some important contents like pdf files or pictures etc. Here, the PageRank of node X_6 is very low when you compare with the other nodes because it's outlinks are being adjusted when we removed the dangling page X_5. So X_6 rank is affected because it is the neighboring node of the dangling page X_5. There are two things happening when we remove the dangling pages during PageRank computation. One is the dangling pages are completely omitted in the ranking which is not correct and the second one is removing the dangling pages affects the PageRank of the neighboring nodes. These two things are being taken care in our proposed method.

Case 2

In the experiment case 2, which is our proposed method, we included the dangling pages X_4 and X_5 in the computation, apart from including a hypothetical node and make all the dangling nodes to point to that hypothetical node. The PageRank computation is shown as a chart in figure 5.

The chart in figure 5 shows the implementation of our proposed method. In this all the dangling pages are included and a hypothetical node called X_h is also included. Here, all the nodes including the dangling nodes are included in the computation and the ranking of all the nodes are shown. The hypothetical node X_h is having a high PageRank value and it will it will be ignored.

The presence of the dangling pages can cause philosophical, storage and computational problems for PageRank. The philosophical problems refer to the removal of the dangling pages during the computation of the PageRank and then add them back after the convergence of the PageRank (in the Brin and Page original PageRank algorithm). Some dangling nodes may get high PageRank and the removal of those nodes are not justified. For example, a quality pdf file may have many in-links from good sources and it should receive a high rank, but in the original PageRank computation it is being ignored. It is not justice on that dangling page point of view.

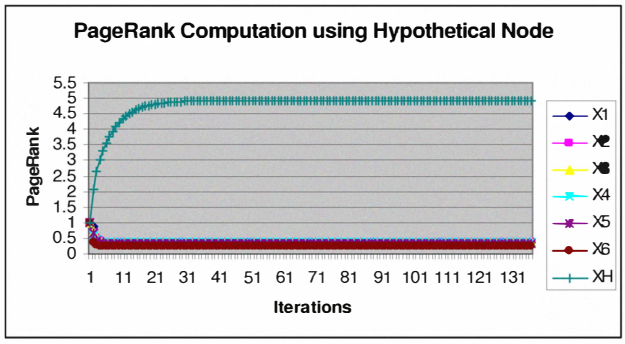

Figure 5 PageRank computation chart using a hypothetical node

The computational problem refers to the rate of convergence and the number of iterations. Incorporating the dangling pages adds only a little computational effort but provides a accurate computation of PageRank. In our experiment case 1 i.e. when we remove the dangling pages, there are only 54 iterations for the PageRank to get it converged. Even though the number of iterations are less comparing to case 2, the PageRanks are not accurate in case 1. In case 2 i.e. using a hypothetical node, the number of iterations are more but the PageRanks are more accurate. In our method, the PageRank get converged almost at the end of 137^{th} iteration. Actually, it is converged at the end of 20^{th} iteration because of the hypothetical node the iterations went up to 137. Nowadays with high performance machines the storage and computational problems can be compromised as long as the algorithms produce accurate ranks. Our method solves the philosophical issues related to the dangling pages by including all the dangling pages in the ranking.

V. CONCLUSION

This paper covers the problems associated with dangling pages and the previous significant work done in handling the dangling pages. We proposed a hypothetical node to handle the dangling pages and the simulation results are shown. Even though our proposed method takes more number of iterations but it produces more accurate ranking of pages.

Most of the Web ranking algorithms are kept as trade secret due to competition. So it is difficult to know how the ranking algorithms are implemented in real but with our limited resources, we implemented the PageRank algorithm and handled the dangling pages with the best of our knowledge.

The future work will be how to reduce the computation (iterations) when the hypothetical node is included and how the matrix computation can be divided into different categories like computing the dangling nodes and non-dangling nodes separately.

REFERENCES

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank Citation Ranking: Bringing order to the Web". *Technical Report, Stanford Digital Libraries* SIDL-WP-1999-0120, 1999.

[2] N. Eiron, K. S. McCurley and J. A. Tomlin, "Ranking the Web Frontier", *Proc. of the WWW Conference*, pp. 309-318, 2004.

[3] A. N. Langville and C. D. Meyer, "A survey of eigenvector methods of web informationretrieval", *Proc of the SIAM*, Vol. 47, No. 1, pp. 135—161, 2005.

[4] M. Bianchini, M. Gori and F. Scarselli, "Inside PageRank". *Proc. of the ACM Transactions on Internet Technology*, Vol. 5, No. 1, pp. 92-128, 2005.

[5] X. Wang, T. Tao, J. T. Sun, A. Shakery and C. Zhai, "DirichletRank: Solving the Zero-One-Gap Problem of PageRank". *Proc. of the ACM Transactions on Information Systems*, Vol. 26, No. 2, Article 10, 2008.

[6] S. D. Kamvar, T. H. Haveliwala, C. D. Manning and G. H. Golub, "Exploiting the Block Structure of the Web for Computing PageRank", *Stanford University Technical Report*, 2003.

[7] S. Brin, R. Motwani, L. Page and T. Winograd, "What can you do with a Web in your pocket?", *Data Engineering Bulletin*, Vol. 21, No. 2, pp. 37-47, 1998.

[8] T. Haveliwala, "Efficient Computation of PageRank", *Stanford University Technical report*, 1999.

[9] C. Pan-Chi Lee, G. H. Golub and S. A. Zenios, "A Fast Two-stage Algorithm for Computing PageRank and its Extensions". *Technical Report SCCM*-2003-15, *Scientific Computation and Computational Mathematics*, Stanford University, 2003.

[10] J. R. Norris, "Markov Chains", *Cambridge University Press,* pp. 1-4, 1996.

[11] Eugene Garfield, "Citation Analysis as a tool in journal evaluation", *Science* 178, pp. 471-479, 1972.

[12] G. Pinski and F. Narin, "Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics", in *Information Processing and Management*, 1976.

[13] S. Chakrabarti, B. Dom, D. Gibson, J.M. Kleinberg, P. Raghavan and S. Rajagopalan, "Automatic resource compilation by analyzing hyperlink structure and association test", *Proc. of the 7th Int. conference on WWW*, pp 65-74, 1998.

[14] R. Lempel and S. Moran, "SALSA: the stochastic approach for link-structure analysis", *ACM Transactions on Information Systems*, pp 387-401, 2000.

Ashutosh Kumar Singh received his PhD degree in Electronics Engineering from Banaras Hindu University, India, in 2000. Presently he is the Head of the Department of Electrical and Computer Engineering, School of Engineering and Science, Curtin University of Technology, Miri, Malaysia. His research interests include verification, synthesis, design, and testing of digital circuits, Web technology and Teaching & Learning. He has published more than 60 research papers to date in different conferences and journals in these areas. He is a co-author of two books, Digital Systems Fundamentals and Computer System Organization and Architecture (Prentice Hall). He has more than 10 years research and teaching experience in various Universities of the India, UK, and Malaysia. He was a member of the editorial board of the University Tun Abdul Razzak (UNITAR) e-journal and has also been involved in the reviewing process of different journals and conferences, such as the IEEE Transactions on Computers, IEEE International Test Conference (ITC), International Conference on Advanced Computing and Communication (ADCOM), and so forth. He is the recipient of the Merit Award from the Institute of Engineers in 2003, the Best Poster Presenter Award from the 86th Indian Science Congress in 1999, and the Best Paper Presenter from the 23rd National Systems Conference (NSC '99) in India.

Ravi Kumar P received his master's degree in Computer Science & Engineering from Anna University, India in 1993. Presently he is a Senior Technical Instructor, Jefri Bolkiah College of Engineering, Kuala Belait, Brunei. His area of interest is Web mining and Database. He is currently doing his Ph. D at Curtin University of Technology, Miri, Malaysia under supervision of Dr. Ashutosh K Singh.

Goh Kwang Leng, Alex is a Computer Science final year student 2010 at Curtin University of Technology, Sarawak Campus, Miri, Malaysia.