페이지랭크 알고리즘의 재검토 : 놈-누수 현상과 해결 방법

(Revisiting PageRank Computation: Norm-leak and Solution)

김 성 진 † 이 상 호 ††
(Sung Jin Kim) (Sang Ho Lee)

요 약 페이지랭크 알고리즘은 웹 문서들을 효과적으로 랭킹(ranking)하는 것으로 알려져 있다. 페이지랭크 알고리즘은 그 유용함에도 불구하고 경우에 따라 문서의 페이지랭크 값을 본래 값보다 작게 계산하는 현상을 유발한다. 본 논문에서는 이러한 현상을 놈-누수(norm-leak)라 명명하고, 웹 문서의 페이지랭크 값을 정확히 산출하는 개선된 페이지랭크 알고리즘과 효율적인 구현방법을 제시한다. 또한, 약 67,000,000개의 실제 웹 문서들에 기존의 페이지랭크 알고리즘과 개선된 페이지랭크 알고리즘을 적용하여 그 결과를 비교 평가한다.

키워드 : 정보검색, 랭킹, 페이지랭크 알고리즘

Abstract Since introduction of the PageRank technique, it is known that it ranks web pages effectively. In spite of its usefulness, we found a computational drawback, which we call norm-leak, that PageRank values become smaller than they should be in some cases. We present an improved PageRank algorithm that computes the PageRank values of the web pages correctly as well as its efficient implementation. Experimental results, in which over 67 million real web pages are used, are also presented.

Key words : Information Retrieval, Ranking, and The PageRank Algorithm

1. Introduction

The link structure of the web is a rich source of information about the content of the environment [1]. The PageRank[2,3] exploits the link structure of web pages to measure the relative importance of web pages. A page has a high PageRank value if there are many pages that point to it, or if pages that point to it have high PageRank values. It is known that PageRank helps rank web pages effectively.

In the computation of the PageRank algorithms [3,4], the hyperlink structure of the web and PageRank values are represented as a matrix and a vector, respectively. A repeated computation of matrix-vector multiplications derives the PageRank vector. The sum of all PageRank values should be maintained to be one during the entire computation. However, we learned that the sum becomes less than one as the computation process continues in some cases. In those cases, all PageRank values become smaller than they should be.

Several studies [4-7] pointed out the drawback of the PageRank algorithm. However, they dealt with the drawback very shortly in terms of theoretical respects only. This paper gives much more detailed descriptions about the drawback and show how severely PageRank values of web pages can be miscalculated in the real web and also in some small artificial webs. We present an improved PageRank algorithm that computes the PageRank values of the web pages correctly. Our algorithm works out well in any situations, and the sum of all PageRank values is always maintained as one.

· This work was supported by Korea Research Foundation Grant. (KRF-2004-005-D00172)
† 학생회원 : 서울대학교 전기컴퓨터공학부
sjkim@oopsla.snu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학부
shlee@comp.ssu.ac.kr
논문접수 : 2004년 9월 3일
심사완료 : 2005년 2월 25일

We also present an efficient implementation for the improved algorithm.

Besides this introduction, section 2 surveys the original PageRank algorithm[4], and section 3 discusses a drawback of the original PageRank algorithm and presents an improved PageRank algorithm. Our experimental evaluations of algorithms and their results are in section 4. Section 5 contains closing remarks.

2. PageRank Computation

Let v be a web page, F_v be the set of pages v points to, and Bv be the set of pages that point to v. Let $N_v = |F_v|$ be the number of unique links from v. The PageRank(PR) equation [3] for v is recursively defined as:

$$PR(v) = \sum_{u \in B_v} \frac{PR(u)}{N_u} \quad (1)$$

The pages and hyperlinks of the web can be viewed as nodes and edges in a directed graph[8]. Let M be a square matrix with the rows and columns corresponding to the directed graph G of the web, assuming all nodes in G have at least one outgoing edge. If there is at least one link from page j to page i(i.e., in other words, more than one link are regarded as one link), then the matrix entry m_{ij} has a value $1/N_j$. The values of all other entries are zero. The PageRank values of all pages are represented as an $N \times 1$ matrix (a vector), Rank. The ith entry, rank(i), in Rank represents the PageRank value of page i.

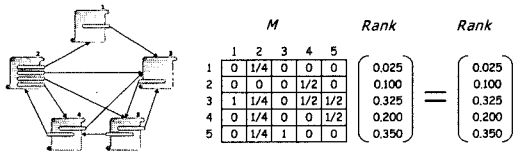


Figure 1 A small web, its matrix, and its PageRank values

Figure 1 shows an example of M and $Rank$. Page 5 has two outgoing links to page 3 and 4 (N_5 = 2), m_{35} and m_{45} of M are $(1/2)$, and m_{15}, m_{25} and m_{55} are 0. Page 5 is pointed by page 2 and 3, so its PageRank value is determined by PageRank values of page 2 and 3. Since page 2 and 3 have four outgoing links and one outgoing link respectively, the PageRank of page 5, rank(5), is the sum of one fourth of rank(2) and rank(3).

Computation of the equation (1) can be represented by a matrix calculation: $Rank = M \times Rank$. The vector, $Rank$, is the principle eigenvector of the matrix M. Rank can be computed by applying M to an initial $Rank$ matrix $[1 / N]_{N\times1}$ repeatedly, where $[1 / N]_{N\times1}$ is an $N \times 1$ matrix in which all entries are $(1/N)$ [3,4]. Let $Rank_i$ be the i^{th} intermediate $Rank$, $Rank_1$ be the initial $Rank$, and $Rank_{i+1}$ be $M \times Rank_i$ (i.e., $Rank_{i+1} = M \times Rank_i$). $Rank_i$ is converged to a fixed point as i increases. The converged $Rank_i$ (i.e., $Rank$) contains all the PageRank values of pages.

To overcome the RankSink[3,4] problem, they introduced a new matrix M', where transition edges of probability (d/N) between every pair of nodes in G are added to M. Those transition edges are called virtual links. Let N be the number of total pages, and $[1 / N]_{N\times N}$ be an $N \times N$ square matrix in which all entries are $(1/N)$. The equation (2) shows matrix M'. The constant d, which is called a dampening factor, should be less than one.

$$M' = (a - d)M + d\left[\frac{1}{N}\right]_{N\times N} \quad (2)$$

The definition of M' has an intuitive basis in random walks on graphs. A surfer keeps clicking on successive links at random, but the surfer gets bored periodically and jumps to a random page. The dampening factor in equation (2) denotes the probability that a surfer jumps to a random page during surfing.

3. An Improved PageRank Computation

An L_1 norm (simply norm) represents the sum of all entries in a vector. A page with no outgoing edge is called a dangling page. Consider the web

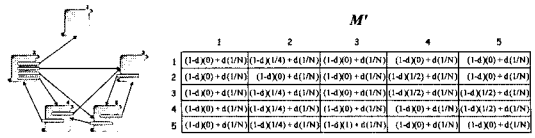


Figure 2 A small web with a dangling page and its matrix M'

$$M' \times \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{pmatrix}_{Rank_i} = \begin{pmatrix} m_{11}\alpha + m_{12}\beta + m_{13}\gamma + m_{14}\delta + m_{15}\varepsilon \\ m_{21}\alpha + m_{22}\beta + m_{23}\gamma + m_{24}\delta + m_{25}\varepsilon \\ m_{31}\alpha + m_{32}\beta + m_{33}\gamma + m_{34}\delta + m_{35}\varepsilon \\ m_{41}\alpha + m_{42}\beta + m_{43}\gamma + m_{44}\delta + m_{45}\varepsilon \\ m_{51}\alpha + m_{52}\beta + m_{53}\gamma + m_{54}\delta + m_{55}\varepsilon \end{pmatrix}_{Rank_{i+1}}$$

The norm of $Rank_{i+1}$ $= (m_{11} + m_{21} + m_{31} + m_{41} + m_{51})\alpha + (m_{12} + m_{22} + m_{32} + m_{42} + m_{52})\beta + (m_{13} + m_{23} + m_{33} + m_{43} + m_{53})\gamma + (m_{14} + m_{24} + m_{34} + m_{44} + m_{54})\delta + (m_{15} + m_{25} + m_{35} + m_{45} + m_{55})\varepsilon$

Figure 3 The norm of $Rank_{i+1}$, given Figure 2

structure and M' in Figure 2. Since page 1 is a dangling page, the norm of the first column vector of M' is not one, but d. The other columns of M' have the norm with one.

Let Ranki be $(\alpha,\beta,\gamma,\delta,\varepsilon)^T$, where T stands for transposition. Then $Rank_{i+1}$ and its norm are represented in Figure 3. The norm of $Rank_{i+1}$ is $(d*\alpha+\beta+\gamma+\delta+\varepsilon)$. Because d is less than one, the norm of $Rank_{i+1}$ is less than the norm of $Rank_{i+1}$ by $(1-d)*\alpha$. This is contrary to the property that the norm of *Rank* should be maintained to be one during the entire computation process. During the iteration of matrix-vector multiplication, $Rank_i$ loses a part of its norm continuously. We call this phenomenon norm-leak. This phenomenon takes place when there are dangling pages.

The norm-leak phenomenon has critical implications in terms of computation. First, PageRank values are likely to be smaller than they should be, and might become all zeroes in the worst case. Second, the iteration process might not converge to a fixed point, because the norms of Ranki and $Rank_{i+1}$ are not the same.

In order to put an emphasis on the original link structure of the web (ignoring the virtual links at the same time) in the computation of PageRank values, we need to use a small value of the dampening factor and use a large number of iterations. Interestingly enough, the norm-leak problem becomes evident when we use a small value of the dampening factor and a large number of iterations. The norm-leak problem is subject to diminish the importance of the original link structure in the computation.

Let us describe an improved PageRank computation. First, we need to define a new matrix M^+. The matrix M^+ is the same as M, except that a dangling column of M is replaced by $[1 / N]_{N\times 1}$. Let D be a set of dangling pages. Let $<1 / N>_{N\times N,p}$ be an $N \times N$ matrix in which entry m_{ij} is $(1/N)$ if j is equal to p and m_{ij} is zero otherwise. The p^{th} column of $<1 / N>_{N\times N,p}$ is exactly $[1 / N]_{N\times 1}$. Then M^+ is expressed as:

$$M^+ = M + \sum_{p\in D} <\frac{1}{N}>_{N\times N,\,p} \quad (3)$$

When a web page has outgoing edges to each of all nodes including itself, then the page is said to have a complete set of edges. Note that each dangling page has a complete set of edges in M^+.

Now, apply the dampening factor to avoid the RankSink problem. We get a new matrix, which we call M^*

$$M^* = (1-d)M^+ + d[\frac{1}{N}]_{N\times N} \quad (4)$$

Given Figure 2, the matrix M^* is described in Figure 4. The first column of M^* can be represented as $[1 / N]_{N\times 1}$, no matter what a dampening factor d is. Note that the norm of each column of M^* is always one.

In the matrixes M' and M^*, each page in G has its own links and a complete set of edges. When a page in both M' and M^* is non-dangling, it distributes d of its importance to all pages, and $(1-d)$ of its importance to pages along with original links. However, a dangling page in M^* evenly distributes all of its importance to all pages, while a dangling page in M' distributes $(1-d)$ of importance to all pages.

	1	2	3	4	5
1	(1-d)(1/N) + d(1/N)	(1-d)(1/4) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(0) + d(1/N)
2	(1-d)(1/N) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(1/2) + d(1/N)	(1-d)(0) + d(1/N)
3	(1-d)(1/N) + d(1/N)	(1-d)(1/4) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(1/2) + d(1/N)	(1-d)(1/2) + d(1/N)
4	(1-d)(1/N) + d(1/N)	(1-d)(1/4) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(1/2) + d(1/N)
5	(1-d)(1/N) + d(1/N)	(1-d)(1/4) + d(1/N)	(1-d)(1) + d(1/N)	(1-d)(0) + d(1/N)	(1-d)(0) + d(1/N)

Figure 4 Example of M^*

The improved PageRank algorithm uses the matrix M^* instead of M' (or M). M^* is not sparse, and handling M^* as it is requires significant overhead in terms of space and time. We transform the expression, $M^* \times Rank$, into an efficient form (see equation (5)), which is more amenable to

efficient processing. M^* is replaced by the right-hand side of equation (4). Multiplication of a matrix, $[\ 1\ /\ N]_{N\times N}$, and a vector, *Rank*, always produces a vector, all of whose elements are *(1/N)*. M^+ is replaced by the right-hand side of equation (3). Finally, multiplication of a matrix, $\sum_{p\in D}<1/N>_{N\times N,p}$ and a vector, Rank, is equivalent to multiplication of a vector, $[1/N]_{N\times N}$, and a constant, $\sum_{p\in D} rank(p)$.

$$M^* \times Rank = \left((1-d)M^+ + d[\frac{1}{N}]_{N\times N}\right) \times Rank$$

$$= (1-d)M^+ \times Rank \ + \ d[\frac{1}{N}]_{N\times 1}$$

$$= (1-d)\left(M + \sum_{p\in D}<\frac{1}{N}>_{N\times N,p}\right)\times Rank \ + \ d[\frac{1}{N}]_{N\times 1}$$

$$= (1-d)M\times Rank \ + \ (1-d)\left(\sum_{p\in D}<\frac{1}{N}>_{N\times N,p} \times Rank\right) \ + \ d[\frac{1}{N}]_{N\times 1}$$

$$= (1-d)M\times Rank \ + \ (1-d)\left([\frac{1}{N}]_{N\times 1} \times \sum_{p\in D} rank(p)\right) \ + \ d[\frac{1}{N}]_{N\times 1}$$

```
1  ∀s, SourceVector[s] = 1 / N          // vector initialization
2  residual = 1
3  while (residual >   ) {
4      ∀d, DestinationVector[d] = 0  // vector initialization
5      leakedValue = 0                  // variable for preserving leaked values
6
7      // 'eof()' is a function returning 'TRUE' when the file pointer indicates the end of the Links file
8      while (not Links.eof()) {
9
10         // source: identification number for source node
11         // n: number of outgoing edges in current source node
12         // dest_i: identification number for i^th destination node
13         Links.read(source, n, dest_1, dest_2, ..., dest_n)
14
15         if n == 0 then
16             leakedValue = leakedValue + SourceVector[source]
17         else
18             // Distribute the PageRank value of the current node n to each destination node
19             for j = 1 ... n
20                 DestinationVector[dest_j] = DestinationVector[dest_j] + (SourceVector[source] / n)
21         End if
22     }
23
24     for k = 1 ... N
25         DestinationVector[dest_k] = DestinationVector[dest_k] + leakedValue / N
26
27     ∀d, DestinationVector[d] = (DAMPENING × DestinationVector[d])  +  ((1 - DAMPENING) / N)
28     residual = ||(SourceVector- DestinationVector)||   // the sum of all the elements in the result vector
29     SourceVector = DestinationVector
30 }
```

Figure 5 Matrix-vector multiplication for the improved PageRank algorithm

In view of implementation issues, it is better to compute PageRank values with equation (5). Only non-zero entries in the matrix M and all entries in the vector *Rank* are stored in disk. For the computation of *(1-d)* M × *Rank*, it is sufficient to read the matrix and the vector only once. The computation time of M × *Rank* is much faster than that of M^* × *Rank* because M is generally sparse. All *(1-d) rank(p)* (the leaking values) is accumulated to a variable during the multiplication of the matrix and the vector, and then the sum of leaking values is redistributed to all pages at once.

The link structure for the web graph should be stored in a file, referred to as Links. The file is separated into three fields: *SourceNode, OutDegree, DestinationNodes*. Suppose node 100 has two outgoing edges to node 200 and 300. Then, a record in the file contains 100 for *SourceNode*, 2 for *OutDegree*, 200 and 300 for *DestinationNodes*. The matrix-vector multiplication can be implemented as shown in Figure 5. We create two arrays of floating point values representing the rank vectors, called *SourceVector* and *DestinationVector*. Each vector has N entries. The PageRank values for iteration i are held in *SourceVector*, and the PageRank values for iteration *(i+1)* are constructed in *DestinationVector*. In the 5th line, leakedValue is initialized to *0* before every matrix-vector multiplication. In the 16th line, for a single multiplication, *leaked values* from dangling pages are accumulated to *leakedValue*. Finally, the accumulated leaked value is distributed to all the pages uniformly, in the 24th and the 25th line, after the multiplication. The dampening factor is denoted as *DAMPENING* in the 27th line.

4. Evaluations

In order to see how severely the norm-leak problem affects the computation of PageRank, we applied the improved PageRank algorithm and the original one with two set of web pages. In our experiment, the maximum number of iteration for computation was set to 15, and multiple dampening factors were considered.

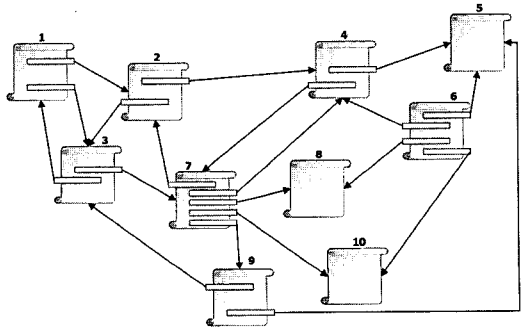
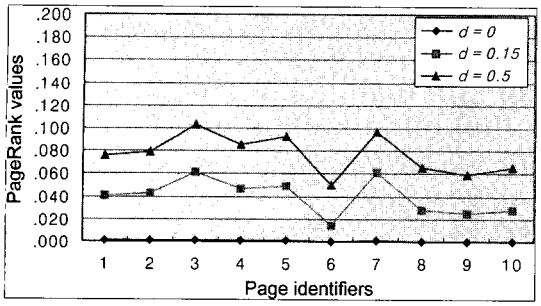
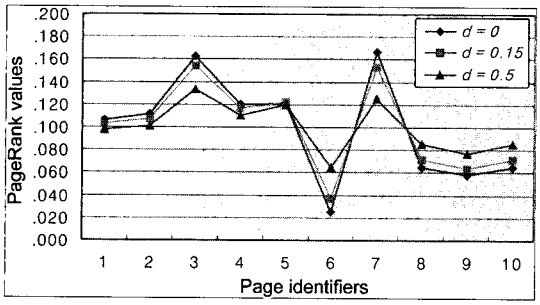


Figure 6 A ten-page web with three dangling pages



(a)

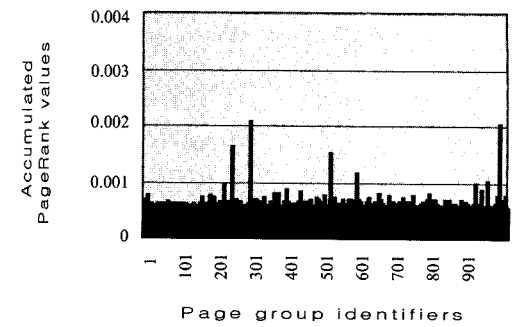


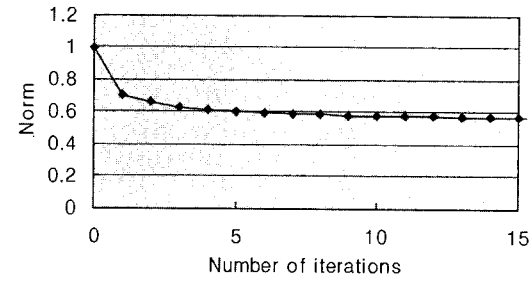
(b)

Figure 7 PageRank values of ten pages

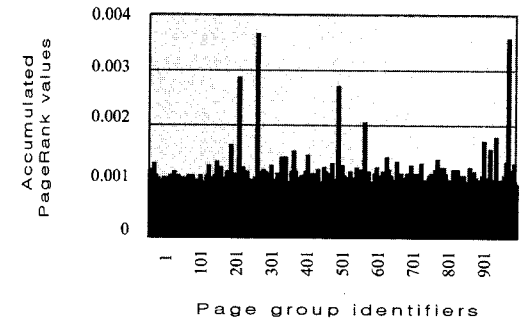
First, we constructed a small web with ten pages, three of which are dangling ones, shown in Figure 6. Figure 7(a) shows the result of the original PageRank algorithm. Because pages 5, 8, and 10 give away a fraction (here d) of their importance to all pages, $(1-d)$ of the importance is lost on each iteration. The smaller a dampening factor value is, the smaller PageRank values are. In particular, the broken line with $d = 0$ does not indicate importance of pages at all, since all PageRank values become virtually zero. Figure 7(b) shows the result of the improved PageRank algorithm. Although there are three dangling pages, the norm of Rank is always maintained as one, independent of dampening factors.

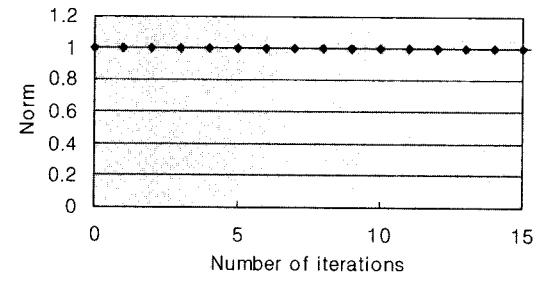
Second, with the help of a web robot [9] we crawled over 67 million real web pages from Korean sites in September 2003, and applied the improved algorithm to them. It is simply not feasible to show all PageRank values of the pages. In order to show the distribution of values of PageRank among the pages, we did as follows. We grouped the pages into 1000 groups randomly. With





(a)





(b)

Figure 8 Accumulated PageRank values and variations of norm (with d = 0.15)

over 67 million pages, a single group contained

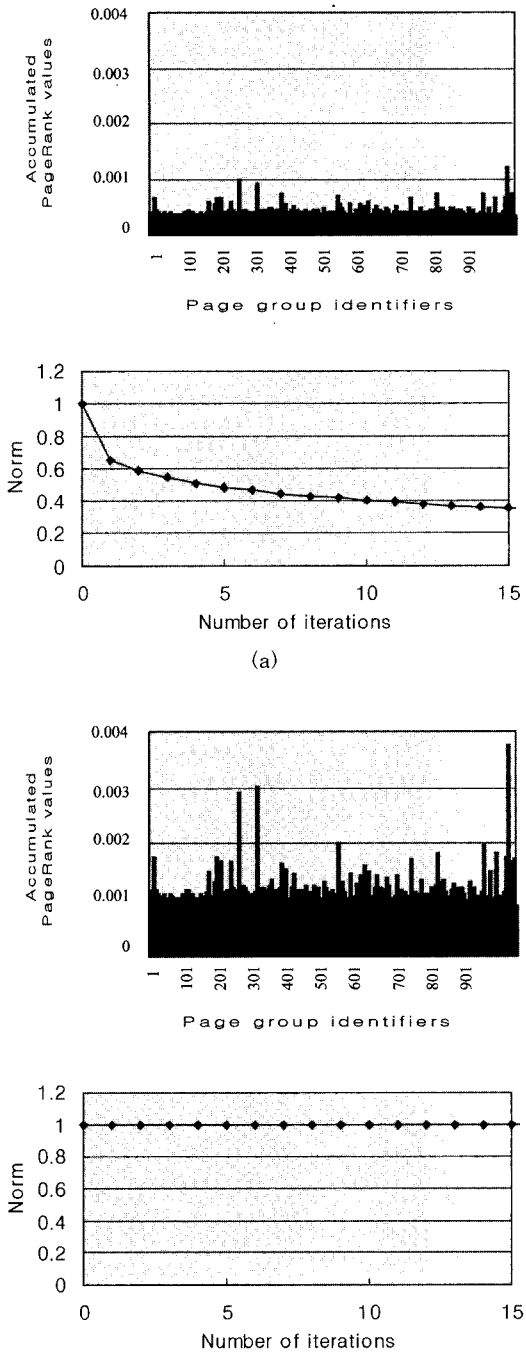


Figure 9 Accumulated PageRank values and variations of norm (with d = 0)

approximately over 67,000 pages. The PageRank values of the pages that belonged to the same group were accumulated into a variable. There were 1000 groups, each of which has the accumulated PageRank value. We plotted all the 1000 accumulated PageRank values graphically, as shown in Figure 8 and 9.

Figure 8(a) shows the experimental result of the original PageRank algorithm with dampening factor 0.15. The upper figure in Figure 8(a) shows that most of the accumulated PageRank values were smaller than 0.001, which implies intuitively that the sum of all values might be less than one. The lower figure in Figure 8(a) shows how the norm changes over iterations. After 15 multiplications, the sum of all PageRank values was 0.57. Figure 8(b) exhibits the experimental result of the improved PageRank algorithm. Most of the accumulated PageRank values were plotted around 0.001. The sum of all PageRank values was indeed exactly one at all times.

Experimental results using a dampening factor zero are shown in Figure 9. In the original PageRank algorithm, a matrix M^* with a dampening factor zero is the same as a matrix M. During the matrix-vector multiplication, the matrix M caused the norm to leak severely, as shown in Figure 9(a). After 15 multiplications, the sum of all PageRank values was 0.35. In the improved algorithm, dropping a dampening factor to zero didn't affect a norm. The upper figures in Figure 8(b) and 9(b) are similar. The norm-leak phenomenon didn't take place in the improved algorithm, as expected.

5. Closing Remarks

The original PageRank algorithm postulates all web pages have at least one hyperlink. The assumption is simply invalid in a practical sense, since there are many dangling pages in the world. We identify the norm-leak problem, which is caused by dangling pages in the web, and also its bad effect on the computation of PageRank values of the web pages. A solution to avoid the norm-leak problem is proposed, and experiments of the algorithms are presented too.

The calculation of PageRank values was done in a modern personal computer (PentiumIV-1.7GHz, 512M bytes main memory). Even in the case of

over 67 million web pages, it took only a few hours to compute PageRank values. In terms of space overhead, our implementation required only a few additional mega-byte disk spaces to store the information on dangling pages.

References

[1] J. Kleinberg, Authoritative Sources in a Hyperlinked Environment, In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, pages 604-632.

[2] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, In Proceedings of World Wide Web Conference, 1998, pages 107-117.

[3] L. Page, S. Brin, R. Motwani, and T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web, unpublished manuscript, Stanford University, 1998.

[4] T. H. Haveliwala, Efficient Computation of PageRank, unpublished manuscript, Stanford University, 1999.

[5] S. J. Kim and S. H. Lee, An Improved Computation of the PageRank Algorithm. In Proceedings of the 24th BCS-IRSG European Colloquium on IR Research, 2000, pages 73-85.

[6] A. N. Langville and C. D. Meyer, Deeper Inside PageRank, Journal of Internet Mathematics, to appear, 2004.

[7] A. Y. Ng, A. X. Zheng, and M. I. Jordan, Stable Algorithms for Link Analysis, In Proceedings of the 24th ACM SIGIR Conference, 2001, pages 258-266.

[8] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, The Web as a Graph: Measurements, Models and Methods, In Proceedings of 5th Computing and Combinatorics Conference, 1999, pages 1-17.

[9] S. J. Kim and S. H. Lee, Implementation of a Web Robot and Statistics on the Korean Web, In Proceedings of the 2nd Human.Society@Internet Conference, 2003, pages 341-350.

김 성 진

1998년 숭실대학교 소프트웨어 공학과 업(학사). 2000년 숭실대학교 대학원 컴퓨터학사(석사). 2004년 숭실대학교 컴퓨터학과 대학원(박사). 2004년~현재 서울대학교 제어계측신기술연구소 연구원. 관심분야는 인터넷 데이타베이스, 데이타베이스 시스템 성능평가

이 상 호

1984년 서울대학교 전산공학과 졸업(학사). 1986년 미국 노스웨스턴대 전산학과(석사). 1989년 미국 노스웨스턴대 전산학과(박사). 1990년~1992년 한국전자통신 연구원. 선임연구원. 1999년~2000년 미국 조지 메이슨대 소프트웨어 정보 공학과 교환 교수. 1992년~현재 숭실대학교 컴퓨터학부 부교수. 관심분야는 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가 및 튜닝