An Improved Computation of the PageRank Algorithm[1]

Sung Jin Kim and Sang Ho Lee

School of Computing, Soongsil University, Korea
1ace@nowuri.net, shlee@computing.ssu.ac.kr
http://orion.soongsil.ac.kr/

Abstract. The Google search site (http://www.google.com) exploits the link structure of the Web to measure the relative importance of Web pages. The ranking method implemented in Google is called PageRank [3]. The sum of all PageRank values should be one. However, we notice that the sum becomes less than one in some cases. 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 to be one. We also present implementation issues of the improved algorithm. Experimental evaluation is carried out and the results are also discussed.

1. Introduction

Web information retrieval tools typically make use of the text on the Web pages as well as the links of the Web pages that contain valuable information implicitly. The link structure of the Web represents a considerable amount of latent human annotation, and thus offers a starting point for structural studies of the Web. Recent work in the Web search area has recognized that the hyperlink structure of the Web is very valuable for locating information [1, 2, 3, 4, 5, 9, 13, 14, 15].

The Google search site (http://www.google.com), which emerged in 1998, exploits the link structure of the Web to measure the relative importance of Web pages. The ranking method implemented in Google is called PageRank [3]. PageRank is an objective measure of citation importance that corresponds with people's subjective idea of importance. Pages that are well cited from many places are worth looking at. PageRank postulates that a link from page u to v implies the author of u recommends to take a look at page v. A page has a high PageRank value if there are many pages that point to it, or if there are pages that point to it and have a high PageRank value. It is known that PageRank helps to rank pages effectively in the Google site.

The PageRank algorithm and implementation details are described in [7, 12]. The PageRank algorithm represents the structure of the Web as a matrix, and PageRank values as a vector. The PageRank vector is derived by computing matrix-vector multiplications repeatedly. The sum of all PageRank values should be one during the 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.

[1] This work was supported by grant No. (R-01-2000-00403) from the Korea Science and Engineering Foundation.

F. Crestani, M. Girolami, and C.J. van Rijsbergen (Eds.): ECIR 2002, LNCS 2291, pp. 73-85, 2002.

In this paper, 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 to be one. We also present implementation issues of the improved algorithm. Experimental evaluation is carried out and the results are also discussed.

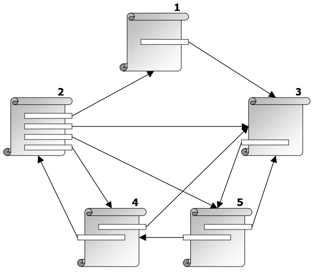
This paper is organized as follows. The PageRank algorithm is presented in section 2. Section 3 identifies drawbacks of the original PageRank algorithm [7] and presents an improved PageRank algorithm. In section 4, we discuss experimental evaluation of algorithms and its results. Section 5 contains closing remarks.

2. PageRank Computation

Let v be a Web page, F_v be the set of pages v points to, and B_v be the set of pages that point to v. Let $N_v = |F_v|$ be the number of links from v. The PageRank (PR) equation [12] for v is recursively defined as:

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

The pages and hyperlinks of the Web can be viewed as nodes and edges in a directed graph [10]. 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 a link from page j to page i, then the matrix entry m_{ij} has a value $1/N_{j.}$ The values of all other entries are zero. PageRank values of all pages are represented as an $N \times 1$ matrix (a vector), *Rank*. The i^{th} entry, *rank(i),* in *Rank* represents the PageRank value of page i.



$$\overset{M}{\begin{array}{c|c|c|c|c|c|} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1/4 & 0 & 0 & 0 \\ \hline 2 & 0 & 0 & 0 & 1/2 & 0 \\ \hline 3 & 1 & 1/4 & 0 & 1/2 & 1/2 \\ \hline 4 & 0 & 1/4 & 0 & 0 & 1/2 \\ \hline 5 & 0 & 1/4 & 1 & 0 & 0 \\ \hline \end{array}} \ \overset{Rank}{\begin{pmatrix} 0.025 \\ 0.100 \\ 0.325 \\ 0.200 \\ 0.350 \end{pmatrix}} = \overset{Rank}{\begin{pmatrix} 0.025 \\ 0.100 \\ 0.325 \\ 0.200 \\ 0.350 \end{pmatrix}}$$

Fig. 1. A small Web, its matrix, and its PageRank values

Fig. 1 shows a simple example of *M* and *Rank*. The rectangular shape like a document denotes a page. A page identifier appears above each page. The small rectangle represents a URL in a page. The directed line denotes a link from one page to another. For an instance, page 5 has two outgoing edges 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 links and one link respectively, the PageRank of page 5, *rank(5),* is the sum of a fourth of *rank(2)* and *rank(3)*. Such computation corresponds to the matrix-vector multiplication.

Computation of the equation (1) can be represented by the following 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 $[1 / N]_{N\times 1}$ repeatedly, where $[1 / N]_{N\times 1}$ is an $N \times 1$ matrix in which all entries are *(1/N)* [7, 12]. 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 PageRank values of all pages.

A vector in which all entries are zero is a zero vector. A page with no outgoing edge is a dangling page. If page *j* is a dangling page, then the j^{th} column of *M* is called a dangling column. A dangling column of *M* is represented as a zero vector. An L_1 norm (simply norm) represents the sum of all entries in a vector. A matrix *M* is irreducible if and only if a directed graph is strongly connected. All columns in an irreducible *M* have norms of value one.

Consider two Web pages that point to each other but to no other pages (i.e., a loop). Suppose there are some Web pages that point to one of them. Then, during the matrix-vector multiplication process, the loop accumulates PageRank values but never distributes any PageRank values (since there are no outgoing edges) [12]. The loop forms a sort of trap, which is called RankSink. Consequently, pages in the loop are likely to have higher PageRank values than they should be.

To overcome the RankSink problem, they [7, 12] introduce another matrix *M'*, where transition edges of probability *(d/N)* between every pair of nodes in *G* are added to *M*. 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 a new matrix *M'*.

$$M' \; = \; (1-d)M + d\,[\frac{1}{N}]_{N\times N} \;. \tag{2}$$

The constant *d* is called a dampening factor, and it is less than one. With Fig. 1, *M'* is constructed as below:

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

The definition of *M'* has an intuitive basis in random walks on graphs. The "random surfer" keeps clicking on successive links at random, but the surfer periodically "gets bored" and jumps to a random page. The probability that the surfer gets bored is a dampening factor.

Even though the matrix *M'* is not sparse, there is no need to store it explicitly. When the equation of $M' \times Rank$ is computed, *M'* is replaced with the right side of the equation (2). The replacement and matrix transformation produce an equation (3). Equation (3) is used to compute PageRank values of all pages.

$$M' \times Rank \; = \; (1-d)M \times Rank + d\,[\frac{1}{N}]_{N\times 1} \;. \tag{3}$$

3. Improved PageRank Computation

We note that there is a drawback of the original PageRank algorithm. Consider the Web structure in Fig. 2. Because there is no outgoing edge in page 1, it becomes a dangling page.

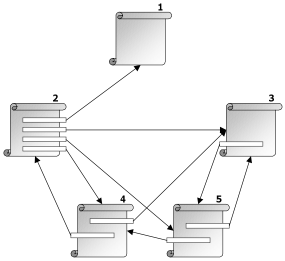


Fig. 2. A small Web with a dangling page

With Fig. 2, *M'* is represented as:

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

Let $Rank_i$ be $(\alpha, \beta, \gamma, \delta, \varepsilon)^T$, *T* stands for transposition. Then $Rank_{i+1}$ and its norm are represented as follows:

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

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

Note that the norm of the first column vector of *M'* (i.e., $m_{11} + m_{21} + m_{31} + m_{41} + m_{51}$) is not one, but *d*, and that all other columns of *M'* have the norm with one. 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 $Rank_i$ by $(1\text{-}d)*\alpha$. 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 zero in the worst case. Second, the iteration process might not converge to a fixed point, because the norms of $Rank_i$ 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 problems caused by the phenomenon become evident when we use a small value of the dampening factor and a large number of iterations. Consequently, the norm-leak problem does not allow us to consider the original link structure significantly in the computation.

In the original Google computation [7, 12], all dangling pages are simply removed from the system, and then the PageRank values are calculated for the remaining Web pages. After that, dangling pages are added back in a heuristic way. This paper presents a simple but elegant technique to solve the norm-leak phenomenon in a detailed level.

3.1 A New Matrix, *M**

Before introducing a new computation technique, we need to define a 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^+ can be expressed as follows:

$$M^{+} \;=\; M + \sum_{p\in D} <\frac{1}{N}>_{N\times N,\,p} \cdot \tag{4}$$

When a node *i* has outgoing edges to each of all nodes including itself, then node *i* is said to have a complete set of edges. The matrix M^+ implies that each dangling page has a complete set of edges, and that a dampening factor is not used. Given Fig. 2 ($N = 5$), M^+ is expressed as:

	1	2	3	4	5
1	1/N	1/4	0	0	0
2	1/N	0	0	1/2	0
3	1/N	1/4	0	1/2	1/2
4	1/N	1/4	0	0	1/2
5	1/N	1/4	1	0	0

Now we are ready to propose a synthesized matrix M^*:

$$M^* = (1-d)M^+ + d\,[\frac{1}{N}]_{N\times N}\ . \qquad (5)$$

Since M^* is based on *M'* and M^+, the matrix M^* can be viewed in two aspects: a dampening factor is applied to M^+, and complete sets of edges are applied to *M'*. A dangling column of M^* is 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. Consequently, it is guaranteed that M^* is irreducible. Given Fig. 2, the matrix M^* is described as:

	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)

Both *M'* and M^* imply that 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.

Now, consider the difference of *M'* and M^* in terms of random surfer model. When a random surfer reaches a dangling page, the surfer in M^* jumps to a page with (*1/N)* probability (note that it is independent of a dampening factor), while the surfer in *M'* jumps to a page with (*d/N)* probability.

3.2 Computational Efficiency

The improved computation for PageRank uses a matrix M^*, which is not sparse. A non-sparse matrix generally requires a large amount of spatial and computational overhead. We describe an efficient computation of $M^* \times Rank$, which does require a little overhead.

In our algorithm, a leaked value of a dangling page should be distributed to all pages additionally. If we distributed the value whenever we found a dangling page, it would not be an efficient approach to compute PageRank values. Instead, we compute the sum of all leaked values from all dangling pages and distribute them to all pages at once. An expression that corresponds to the accumulation and distribution of leaked values needs to be added to equation (3). Equation (6) shows the final equation. On computation of *(1-d) M* × *Rank*, we accumulate all *(1-d) rank(p)* (leaked values). After the computation of *(1-d) M* × *Rank*, the sum is redistributed to all pages.

$$M^{*} \times Rank \;=\; (1-d)M \times Rank \;+\; (1-d)\,[\frac{1}{N}]_{N\times 1} \times \sum_{p\in D} rank(p) \;+\; d\,[\frac{1}{N}]_{N\times 1} \cdot \qquad (6)$$

There is one matrix-vector multiplication in the equation (6). Because the matrix *M* is generally sparse, only non-zero entries in the matrix can be stored and used to compute the matrix-vector multiplication. The matrix *M* should be stored in a disk, because of its huge size. Given Fig. 2, a data structure for *M*, referred to as *Links* (see Fig. 3 [7]), is stored on disk. *Links* is scanned only once for the matrix-vector multiplication.

Source Node	Out Degree	Destination Nodes
2	4	1, 3, 4, 5
3	1	5
4	2	2, 3
5	2	3, 4

Fig. 3. Data structure of *Links*

Information of dangling pages has to be stored in *Links* in our computation. The value of *Destination Nodes* of a dangling page is null, representing all pages. This additional information doesn't affect the matrix-vector multiplication. Given Fig. 2, Fig. 4 shows *Links* for our algorithm.

Source Node	Out Degree	Destination Nodes
1	5	Null
2	4	1, 3, 4, 5
3	1	5
4	2	2, 3
5	2	3, 4

Fig. 4. *Links* for the improved PageRank algorithm

Now consider the size of the additional space for dangling pages. For each dangling page, we need to know *Source Node, Out Degree,* and *Destination Nodes. Source Node* is a sequential number, which does not need to be stored explicitly. The value of *Destination Nodes* is null, which is of length zero. The pure space overhead for *n* dangling pages is _'n * the length of Out Degree',_ which is _(4*n)_ presuming an integer value is represented by 4 bytes.

4. Evaluations

In order to show that the improved PageRank algorithm can solve the norm-leak and the RankSink problem, we implemented and evaluated the improved PageRank algorithm and the original one. We performed two experiments; one with real Web pages and one with a small set of artificial Web pages. The hardware we used was PentiumII-350 with 192MB main memory, running the Accel Linux Version 6.1 operating system.

4.1 An Experiment Using the Real Web

We applied the both algorithms to over 10 million Web pages that had been collected from most of Korean sites. 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 over 10 million, a single group contained approximately over 10,000 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 Fig. 5 and 6. The maximum number of iterations and the dampening factor for computation were set as 30 and 0.15, respectively.

Fig. 5 shows the distribution of the accumulated PageRank values that were derived using the original algorithm. Since there were 1000 groups and most of the accumulated PageRank values were smaller than 0.001, we can inference that the sum of all values might be less than one. The sum of all the PageRank values was indeed 0.52 in our experiment. It shows that the norm-leak phenomenon occurs in the original algorithm.

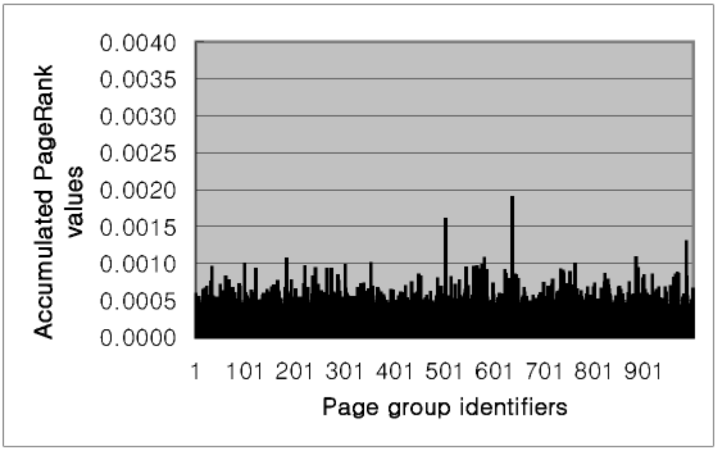


Fig. 5. Accumulated PageRank values by the original algorithm

Fig. 6 shows the distribution of PageRank values that were derived using the proposed algorithm. Note that most of the accumulated PageRank values were plotted around 0.001. The sum of all PageRank values was indeed exactly one in our experiment, as it should be. The norm-leak phenomenon didn't take place in the improved algorithm.

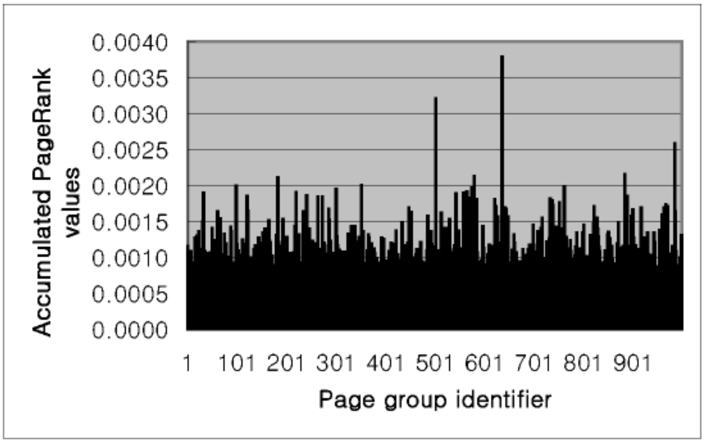


Fig. 6. Accumulated PageRank values by the improved PageRank algorithm

4.2 An Experiment Using Artificial Small Webs

Ten pages are used for our testing. These pages are appropriately linked each other to simulate the RankSink problem and the norm-leak problem. Fig. 7 shows a basic organization of the pages. Our experiments performed under four cases. The maximum number of iteration for computation was set to 20. Three dampening factors were considered.

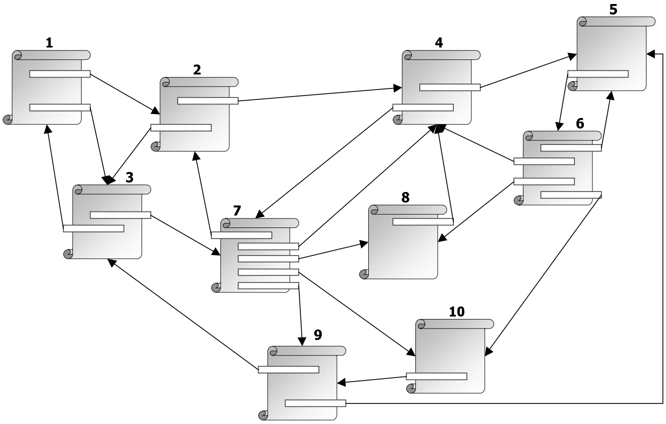


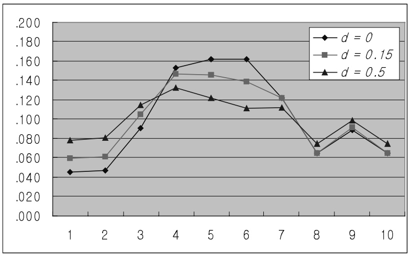
Fig. 7. A basic structure of a ten page Web

(1) Case 1: No RankSink, No Dangling Pages

The basic organization in Fig. 7 is tested. The ten pages have no RankSink and no dangling page. This case serves as a baseline of our testing. Fig. 8 shows the results of the two algorithms. Two algorithms behaved identically in this case, as expected. See the line with *d = 0.* Page 5 and 6 have the highest PageRank value and page 1 has the lowest one. The PageRank value of page 6 is the same as one of page 5, because page 6 is linked by page 5 that has one outgoing edge only. The smaller a dampening factor is, the bigger the difference between the highest PageRank and the lowest PageRank is. The norm-leak phenomenon does not happen in this case, because there is no dangling page. The norm of *Rank* is always one in the both algorithms.

Page ID	*d = 0*	*d = 0.15*	*d = 0.5*
1	.045 403 000	.059 643 500	.078 636 400
2	.047 125 000	.061 079 100	.080 842 000
3	.090 870 700	.105 046 000	.114 545 000
4	.153 046 000	.146 574 000	.132 772 000
5	.161 537 000	.145 503 000	.121 727 000
6	.161 381 000	.138 672 000	.110 863 000
7	.121 940 000	.121 983 000	.111 829 000
8	.064 798 600	.065 199 400	.075 040 800
9	.089 100 000	.091 146 100	.098 703 400
10	.064 798 600	.065 199 400	.075 040 800
Norm	1	1	1
Iterations	20	20	20

(a)

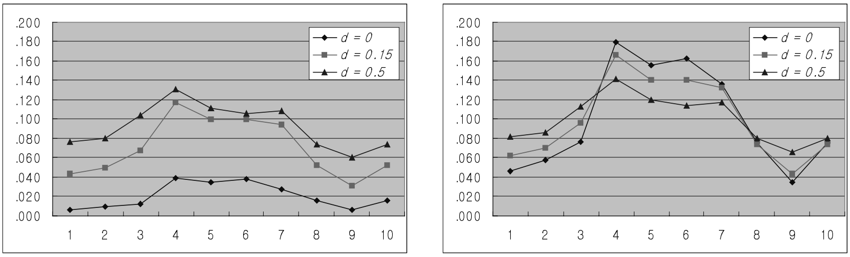


(b)

Fig. 8. Case 1 result

(2) Case 2: No RankSink, One Dangling Page

Suppose that the link from page 10 to 9 is removed in Fig. 7, so that page 10 is a dangling page. Ten pages have no RankSink, but have one dangling page. Case 2 shows that our algorithm can solve the norm-leak phenomenon, which the existing PageRank algorithm cannot solve.



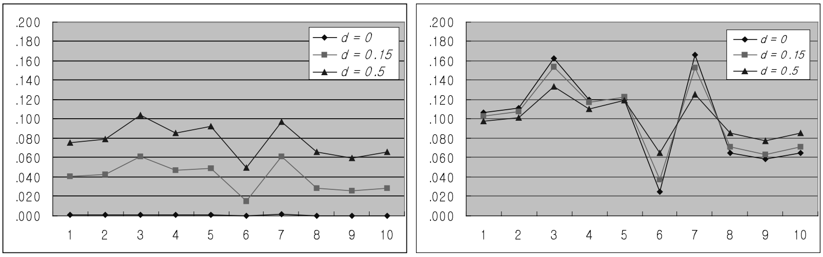
(a) (b)

Fig. 9. Case 2 results

Fig. 9(a) shows the result of the existing PageRank algorithm. Because page 10 confers a fraction (here *d*) of its importance to all pages, *(1-d)* of the importance is lost at each iteration. The norm of *Rank* is always less than one. The smaller a dampening factor value is, the smaller values of PageRank are. If the number of iteration were set to a much higher value, the PageRank values would be much smaller.

Fig. 9(b) shows the result of the improved PageRank algorithm. Although page 10 is a dangling page, the norm of *$Rank_i$* is always kept as one. The norm of *Rank* is one, independent of a dampening factor and the number of iterations. It is interesting to note that the distributions of PageRank values vary depending on dampening values.

(3) Case 3: No RankSink, Three Dangling Pages
Suppose that the links from pages 5, 8, and 10 are removed in 7. Pages 5, 8, and 10 become dangling. Ten pages have three dangling pages and no RankSink. Case 3 shows that our computation exhibits noticeable difference when there are many dangling pages.



(a) (b)

Fig. 10. Case 3 results

Fig. 10(a) shows the result of the existing PageRank algorithm. Since the norm-leak takes place at three dangling pages, PageRank values in Fig. 10(a) are much smaller than those in Fig. 9(a). With many dangling pages, the norm of *Rank* leaks severely. See the broken line with *$d = 0$*. The line cannot tell us importance of pages, since all PageRank values are virtually zero. If we applied many extra iterations of matrix-vector multiplication additionally, the PageRank values with *$d = 0.5$* would converge to zero.

Fig. 10(b) shows the result of the improved PageRank algorithm. Although there are three dangling pages, the norm-leak phenomenon does not happen. Fig. 10 shows that the effect of our algorithm becomes evident as the number of dangling pages increases.

(4) Case 4: RankSink, No Dangling Page
Suppose that the links from pages 2, 3, and 7 to pages 4, 7, and 2 are removed in Fig. 7. Fig. 11 shows the resulting Web. Pages 1, 2, and 3 form a group 1, and the rest of the pages form a group 2. Note that a link from page 9 to page 3, which is an only link from a group 2 to a group 1, exists. Each of the ten pages has at least one outgoing edge, so that there is no dangling page. Case 4 shows that the improved PageRank algorithm handles the RankSink phenomenon, just as the PageRank algorithm does.

The two algorithm works identically. Fig. 12 shows the result. In both algorithms, the norm of *Rank* was always maintained as one. Pages in group 1 have relatively high PageRank values, whereas pages in group 2 have low values. This is due to the fact that pages of group 1 received most of the importance of pages in group 2. We see that the improved PageRank can also solve the RankSink phenomenon by choosing an appropriate *d*.

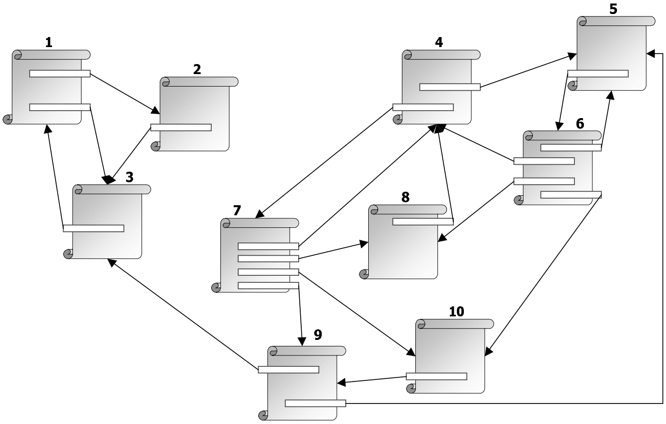


Fig. 11. A RankSink structure of a ten page Web

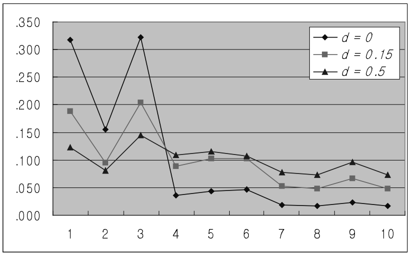


Fig. 12. Case 4 result

5. Conclusion

In the paper, we presented an improved PageRank algorithm that solves the norm leak phenomenon. In the new PageRank algorithm, pages have the right PageRank values and the iteration process always converges to a fixed point. We also described efficient implementation issues of our algorithm. The implementation of our algorithm does not require a large amount of spatial and computational overhead. Recently, we learned that our approach had been very briefly mentioned in [7, 11]. However, the description of this paper about the problem is much more detailed than those of [7, 11] are, and implementation issues were also discussed in this paper.

We applied our algorithm to over 10 million Web pages that had been collected from most of Korean sites. The elapsed times of the both algorithms to compute PageRank values were less than an hour in a medium-sized server machine. Our algorithm only needs a few mega-byte disk spaces to store the information of dangling pages additionally. It took less than minutes to read the additional space.

We have learned that a number of pre-processing operations are recommended to compute the PageRank values correctly. These operations may affect PageRank values and computing time significantly. Examples of these pre-operations include how to extract URLs in Web pages (in particular a URL associated with an image in a script language), how to convert a relative URL to an absolute URL, how to map a URL to a page identifier, and so on. The effectiveness of a Web crawler affects PageRank values significantly, too. In order for the link graph to represent the real Web exactly, a crawler should collect all Web pages. Crawling all Web pages is not a simple job in our experience.

References

1. G. O. Arocena, A. O. Mendelzon, and G. A. Mihaila: Applications of a Web Query Language, Proceedings of WWW6 (1997), 1305-1315
2. K. Bharat and M. Henzinger: Improved Algorithms for Topic Distillation in Hyperlinked Environments, Proceedings of the 21st ACM SIGIR Conference (1998), 104-111
3. S. Brin and L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine, Proceedings of WWW7 (1998), 107-117
4. J. Carriere and R. Kazman: Webquery: Searching and Visualizing the Web through Connectivity, Proceedings of WWW6 (1997), 1257-1267
5. J. Dean and M. R. Henzinger: Finding Related Web Pages in the World Wide Web, Proceedings of WWW8 (1999), 1467-1479
6. D. Gibson, J. Kleinberg, and P. Raghavan: Inferring Web Communities from Link Topology, Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (1998), 225-234
7. T. H. Haveliwala: Efficient Computation of PageRank, unpublished manuscript, Stanford University (1999)
8. E.-J. Im and K. Yelick: Optimizing Sparse Matrix Vector Multiplication on SMPS, Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing (1999), 127-136
9. J. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (1998), 604-632
10. J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins: The Web as a Graph: Measurements, Models and Methods, Invited survey at the International Conference on Combinatorics and Computing (1999), 1-17
11. A. Y. Ng, A. X. Zheng, and M. I. Jordan: Stable Algorithms for Link Analysis, Proceedings of the 24th ACM SIGIR Conference (2001), 258-266
12. L. Page, S. Brin, R. Motwani, and T. Winograd: The PageRank Citation Ranking: Bringing Order to the Web, unpublished manuscript, Stanford University (1998)
13. J. Pitkow and P. Pirolli: Life, Death, and Lawfulness on the Electronic Frontier, Proceedings of the Conference on Human Factors in Computing Systems (CHI 97) (1997), 383-390
14. P. Pirolli, J. Pitkow, and R. Rao: Silk from a Sow’s Ear: Extracting Usable Structures from the Web, Proceedings of the Conference on Human Factors in Computing Systems (CHI 96) (1996), 118-125
15. E. Spertus: ParaSite: Mining Structural Information on the Web, Proceedings of WWW6 (1997), 1205-1215
16. S. Toledo: Improving the Memory-system Performance of Sparse-matrix Vector Multiplication, IBM Journal of Research and Development, volume 41 (1997)
17. Google Search Engine: http://www.google.com