國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用網路中的強連結單元計算個人化網頁排序

Computing Personalized PageRank Using Strongly Connected Components in Web

曾榮國

Rung-Guo Tzeng

指導教授：陳銘憲 博士

Advisor: Ming-Syan Chen, Ph.D.

中華民國 97 年 7 月

July, 2008

Acknowledgements

The finish of this thesis should be attributed to the unconditional help and priceless support of many people. I will always bear them in mind.

At the very first, I'm honored to express my deepest gratitude to my advisor, Prof. Ming-Syan Chen, with whose able guidance I could have worked out this thesis. He has offered me valuable ideas, suggestion, and criticisms with his profound knowledge and rich research experience. I'm much obliged to his effort of pushing me and helping me complete this thesis.

I also cordially express my gratitude to Dr. Chih-Ming Hsu, whose patient guidance and invaluable suggestion are indispensable to this thesis.

Last but not the least, I would like to thank my families and friends for their support, thoughtfulness, and encouragement in all respects from the beginning of my postgraduate study.

Abstract

PageRank is an important ranking technique used in search engines. By using different personalization vectors, search engine companies can compute specific and adaptive PageRank vectors for different classes of users. However, computing even one PageRank vector consumes a lot of time. To address this problem, we propose the SCC (Strongly Connected Component) PageRank algorithm. The main spirit of SCC PageRank is utilizing the old PageRank vector to deduce the new one. We decompose the original linear system of PageRank model into linear subsystems. By using the dependency relation between these linear subsystems, we translate the original computation into a hierarchical manner. While computing personalized PageRank vectors, we check the necessity of recomputation. Thus, we can prevent some iterative recomputations and achieve an improvement of efficiency. As shown by our experimental results, while computing several personalized PageRank vectors, SCC PageRank performs better than the known accelerating algorithms in most cases.

Keywords: Algorithm, Search Engine, Linkage Analysis, Linear System, PageRank

摘要

網頁排序是搜尋引擎中一項重要的排序技術。藉由不同的個人化向量，搜尋引擎公司可以為不同群組的使用者計算出特殊及適合的網頁排序向量。然而，僅計算單一網頁排序向量便需花費許多時間。為了解決這個問題，我們提出了 SCC (Strongly Connected Component)網頁排序演算法。SCC 網頁排序演算法的主要精神在於利用舊的網頁排序向量推導出新的網頁排序向量。我們將網頁排序模組中原先的線性系統拆解為線性子系統。利用這些線性子系統間的相依關係，我們將原本的計算轉變為階層式架構。在計算個人化網頁排序向量時，我們檢查重覆計算的必要性。因此，我們可以避免一些迭代的重覆計算並達到效能上的改進。根據我們的實驗成果顯示，在計算諸多個人化網頁排序向量時，SCC 網頁排序演算法的表現在許多情況下可優於已知的加速演算法。

關鍵字：演算法、搜尋引擎、鏈結分析、線性系統、網頁排序

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid growth of the World Wide Web has brought us a great deal of information. To efficiently retrieve information from this great information warehouse, we need search tools to help us finding queried pages. However, the size of Web results some issues to search engines. Since there are billions of pages on the Web, there may be thousands of pages related to the queried keyword. To fit the users' requirements closely, most search engine companies develop their own criterions to rank and list these resulting pages. In 1998, Page et al developed a celebrated ranking criterion called PageRank [17]. PageRank is query-independent. It globally measures the importance of each page. The importance of all Web pages are computed and saved off-line. In the PageRank model, there exists a special element, the personalization vector. The personalization vector is a probability distribution, which denotes the probability of each page that a surfer may jump to. Initially, the standard personalization vector is an uniform distribution. Later, Page et al suggest nonuniformly distributed personalization vectors. Using nonuniformly distributed personalization vector as the base, the corresponding PageRank vector can be regarded as preferring some Web pages. Thus, the search engine companies can compute specific and adaptive PageRank vectors for different classes of users. The idea of personalized PageRank sounds attractive. However, the misery is that computing even one PageRank vector consumes a lot of time. How to efficiently compute different personalized PageRank vectors is called personalized PageRank problem. To address this problem, much research has been elaborated on this area.

In [10], Jeh and Widom present a scalable personalized PageRank approach. They indicate that there is linear relationship between personalization vectors and their corresponding personalized PageRank vectors. To efficiently compute the personalized PageRank vectors, they precompute several vectors, which Jeh and Widom call basis vectors. The personalized PageRank vector can be expressed as a linear combination of these basis vectors. Depending on the size of the Web, the number of basis vectors can be dynamically adjusted. Thus, at query time, an approximate personalized PageRank vector can be constructed.

Another approach to address the personalized PageRank problem is BlockRank algorithm, which is proposed by Kamvar et al [11]. The BlockRank includes 4 steps: 1. Compute local PageRank for each block (in Kamvar et al's paper, a block is a set of nodes belonging to the same host). 2. Compute the block transition matrix and Block-Ranks 3. Weight each local PageRank and use the transition matrix to compute an approximation to the global PageRank. 4. Use the approximation computed in step 3 as the starting vector and run the standard PageRank iteration. Initially, BlockRank is used for accelerating the computation of PageRank vector. Later, Kamvar et al find that this algorithm can be used in computing personalized PageRank vectors. They make a restriction on personalization vector. Instead of assigning probability to each page, Kamvar et al assign probability to each "host". Thus, when the surfer jumps, he or she chooses a host rather than a page. Under this restriction, each local PageRank stays unchanged while computing different personalized PageRank vectors. Therefore, only the last 3 steps of BlockRank need to be redone.

Obviously, all these prior works accelerate the computation of personalized PageRank. To address the personalized PageRank problem, we propose a different algorithm called SCC (Strongly Connected Component) PageRank. The SCC PageRank is constructed on top of the known fast computing PageRank algorithm, the two-way reordered PageRank [9].

In computing PageRank, the intuitive method, the power method, regards all pages as the same. Thus, the size of the corresponding linear system is the same as the size of Web. To solve this linear system, power method uses iterative computation to solve it. In two-way reordered PageRank, Hsu et al respectively reordered the *reverse-dangling pages* (pages without in-links) and the *dangling pages* (pages without out-links) to the front and rear of the hyperlink matrix. Hsu et al indicated that the computation of these two kinds of pages is straight and needs no iteration. Thus, we can exclude them from the linear system, and reduce the size of the linear system. By exploiting both dangling and reverse-dangling pages, two-way reordered PageRank can efficiently reduce the computational complexity.

We noticed that the computation of reverse-dangling and dangling pages in two-way reordered PageRank proceeds like a *propagation*. That is, we firstly compute the unnormalized PageRanks of some reverse-dangling pages. Then use their results to compute the unnormalized PageRanks of some other reverse-dangling pages, and so on. For dangling pages, the computation is similar. This gives us an idea. Consider that we have computed the unnormalized PageRank vector using uniformly distributed personalization vector. In computing personalized PageRank, if the personalization values of pages, which are at the front of this propagation, are the same as they are in the uniform distribution, we do not need to recompute their unnormalized PageRanks. Thus, we can save some time and get an improvement in performance. This idea sounds feasible. However, the other pages, called *normal pages*, cause a problem. In two-way reordered PageRank, the normal pages are located between reverse-dangling pages and dangling pages. To compute the PageRanks of these normal pages, we need to solve their corresponding linear system. As shown by Hsu et al's experimental results, solving this linear system consumes most of computing time. Thus, the effect of improvement that we have discussed is diluted. To resolve this problem, we get deeper into the linear system, and split it into linear subsystems. By the dependency relation between these linear subsystems, we can solve the linear subsystems one by one. Thus, the computation for normal pages also proceeds like propagation. Combining all kinds of pages, the whole computation proceeds in a fully propagated manner. While computing personalized PageRank, we compare the personalization vector with the uniform distributed one. If the personalization values of the pages, which are at the front of the propagation, stay unchanged, we do not need to recompute their unnormalized PageRanks. At a long run, we can save time in recomputing several linear subsystems. Thus, we can get an improvement in performance.

To ensure that our idea is feasible, we conduct an implementation of personalization vector. We use a concept of *rating score*. We assign a rating score to each Web page, and the corresponding personalization vector is the normalization of the rating vector. With this implementation, we indicate that if the rating scores of some pages, which are at the front of the propagation, stay unchanged, we can quickly compute their new unnormalized PageRanks (i.e. we do not need iteration). Theoretical space and time complexities of the SCC algorithm are analyzed. As shown by our experimental results, the proposed SCC PageRank takes more time than two-way reordered PageRank and power method need while computing the standard PageRank vector. The extra time penalty is mainly caused by the I/O overhead of decomposing the original linear system. While computing personalized PageRank vectors, we do not need to redo the decomposition. Thus, the time is spent only on computation. While computing personalized PageRank vectors, the SCC PageRank is able to outperform the power method. In the worst case (i.e. no recomputation can be prevented), the performance of SCC PageRank is about the same as that of two-way reordered PageRank. In other cases, the SCC PageRank can potentially perform better than two-way reordered PageRank.

The remainder of this thesis is organized as follows. Preliminaries are given in Chapter 2. We develop the SCC PageRank algorithm in Chapter 3. The experimental evaluation and results are discussed in Chapter 4. Chapter 5 compares the SCC PageRank with some prior works. Finally, we present our conclusions in Chapter 6.

Chapter 2

Preliminaries

In this chapter, we present the preliminaries for the Google's PageRank model, two-way reordered PageRank, and the personalized PageRank problem. Symbols used throughout this thesis are listed in Table 2.1.

2.1 Web Model

We use a digraph $G = (V, E)$ to denote the Web. The set V of vertices represents pages in Web. The set E of directed edges represents hyperlinks between pages. Thus, we can use an adjacency matrix, $\mathbf{H}_{n\times n}$, to represent the Web graph, where n is the size of the Web. If the entry, $h_{i,j}$, of $\mathbf{H}$ is nonzero, there is a hyperlink from page i to page j. In the PageRank model, each entry, $h_{i,j}$, of $\mathbf{H}$ is set as follow:

$$h_{i,j} = \begin{cases} 1/out(i), & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases},$$

where $out(i)$ is the out-degree of page i. The matrix $\mathbf{H}$ is nonnegative, sparse and *row normalized* (which means the nonzero rows sum to 1). The positive elements of row i correspond to the out-links of page i, whereas the positive elements of column i correspond to the in-links of page i.

Table 2.1: List of symbols.

Notation	Definition
$\mathbf{H}$	Row normalized hyperlink matrix of Web
$\mathbf{I}$	The identity matrix
$\mathbf{0}$	The zero matrix
$\mathbf{e}$	The column vector of all 1s
$\mathbf{v}$	The personalization vector of Web pages
π	The PageRank vector of Web pages
$\mathbf{u}$	The rating vector of Web pages
α	The damping factor of PageRank model
τ	The convergence tolerance in Jacobi
$\|x\|_1$	The L_1 norm of vector x
$n(M)$	The order of the square matrix M, i.e. the size of M is $n(M) \times n(M)$
$nnz(M)$	The number of nonzeros in matrix M
$out(i)$	The out-degree of page i
$in(i)$	The in-degree of page i
RD-node	A node has no in-links
N-node	A node has both in-links and out-links
D-node	A node has no out-links and has at least one in-link

Hence, the matrix $\mathbf{H}$ contains a zero row for each node without out-links (i.e., a dangling page), and contains a zero column for each node without in-links (i.e., a reverse-dangling page) [1].

2.2 Google's PageRank Model

Here we give a brief introduction to Google's PageRank model. Deeper description can be found in [17] or [13]. We can use a simple sentence, "*a page is important if it is pointed by other important pages*", to express the concept of PageRank. The meaning of this sentence is that the importance of a page i is the sum of the important scores of all pages pointing into page i. Thus,

[1]In this thesis, the terms "node" and "(Web) page" are used interchangeably.

the PageRank value of a page i, denoted π_i, is computed by the following equation [17]

$$\pi_i = \sum_{(j,i)\in E} \frac{\pi_j}{out(j)}. \tag{2.1}$$

Given an initial PageRank to each page, we can iteratively compute Eq. (2.1). When the iteration stops, the result is the final PageRank. By collecting all PageRanks of all Web pages into a row vector, π^T, we can translate Eq. (2.1) into the following linear system.

$$\pi^T = \pi^T \mathbf{H}. \tag{2.2}$$

Note that Eq. (2.2) is like the power method applied to a Markov chain with transition probability matrix $\mathbf{H}$ [13]. Since we hope there is a unique positive PageRank vector for Eq. (2.2), some adjustments are made to Eq. (2.2). In Markov theory, the power method converges to a unique vector if the Markov matrix, P, is *stochastic, irreducible,* and *aperiodic*. The converged vector is called *stationary vector*. The first adjustment creates a matrix $\mathbf{S}$ such that $\mathbf{S} = \mathbf{H} + (1/n\mathbf{a}\mathbf{e}^T)$, where $\mathbf{a}^T = (a_1, a_2, \ldots, a_n)$, $a_i = 1$ if page i is a dangling node and 0 otherwise. This adjustment promises the stochasticity. To be irreducible and aperiodic, the second adjustment creates a matrix $\mathbf{G}$ such that $\mathbf{G} = \alpha\mathbf{S} + (1-\alpha)\mathbf{E}$, where α is a scalar between 0 and 1, and matrix $\mathbf{E} = 1/n\mathbf{e}\mathbf{e}^T$ is the *teleportation matrix*. In Page et al's paper, they use a model of random surfer to interpret these two adjustments. Imagine a Web surfer who surfs the Web by following the hyperlinks in each page. When he enters a new page, he randomly chooses an out-link to go to the next page. When the surfer gets into a dangling page, given no out-links to choose, he teleports to a random page by inputting an URL. Thus, the first adjustment assigns probability to each page the surfer may teleport to when he gets into a dangling page. The second adjustment represents that the surfer has probability of α to follow this surfing model, and probability of (1-α) to randomly teleport between pages. The entry, $\mathbf{E}_{i,j}$, denotes the probability of surfer's teleportation to page j when he is at page i. By these two adjustments, the definition of PageRank vector becomes the solution of the following Markov model ([13], page 37):

$$\pi^T = \pi^T \mathbf{G}, \;\; \pi^T \mathbf{e} = 1. \tag{2.3}$$

2.3 The Personalization Vector $\mathbf{v}^T$

Initially, Page et al used $\mathbf{E} = 1/n\mathbf{e}\mathbf{e}^T$ as the teleportation matrix. This teleportation matrix represents that the surfer teleports to each page with equal probability. Later, Page et al suggested a modification. Instead of using $1/n\mathbf{e}\mathbf{e}^T$, they used $\mathbf{E} = \mathbf{e}\mathbf{v}^T$ as the teleportation matrix. The vector, $\mathbf{v}^T$, is a probability vector called *personalization vector*. In this vector, each entry, $\mathbf{v}_i^T > 0$, represents the teleportation probability of each page i. Thus, different teleportation matrices correspond to different personalization vectors. Since each entry of $\mathbf{v}^T$ is positive, every node still directly connects to every other node. This means that the matrix $\mathbf{G}$ is still irreducible and aperiodic, and thus the unique stationary vector for the Markov chain still exists. Using $\mathbf{v}^T$ instead of $1/n\mathbf{e}^T$ means that the teleportation probabilities are no longer uniformly distributed. Thus, search engine companies have different options in computing PageRank vectors. As mentioned in Chapter 1, each specific personalization vector represents specific interest of surfers in different Web pages.

2.4 The Personalized PageRank Problem

Since the size of Web is large, computing several personalized PageRank vectors consumes lots time. As mentioned in Chapter 1, how to efficiently compute several personalized PageRank vectors is called personalized PageRank problem. In this thesis, our goal is utilizing the old unnormalized PageRank vector to quickly deduce the new unnormalized PageRank vector. We can formalize our goal as: Given the hyperlink matrix $\mathbf{H}$, the old personalization vector $\mathbf{v}$, the old corresponding unnormalized PageRank vector π, and the new personalization vector $\mathbf{v}^*$, find the new corresponding PageRank vector π^*.

2.5 Linear System for PageRank Problem

We can regard Eq. (2.3) as a linear system instead of a Markov chain. The computation of PageRank in Eq. (2.3) is equivalent to the problem of finding the dominant eigenvector of matrix $\mathbf{G}$. Thus, Eq. (2.3) can be rewritten as the solution of the following linear system accompanied with $\pi^T\mathbf{e} = 1$:

$$\pi^T(\mathbf{I} - \alpha\mathbf{S}) = (1 - \alpha)\mathbf{v}^T.$$

Note the coefficient matrix $(\mathbf{I} - \alpha\mathbf{S})$ can be a dense matrix if the number of dangling nodes is large. This is not favorable for computing. To avoid this problem, we use the following theorem to translate the PageRank problem into a sparse linear system formulation.

Theorem 1. ([13], page 73) Let $\hat{\mathbf{x}}$ be the solution of the linear system

$$\mathbf{x}^T(\mathbf{I} - \alpha\mathbf{H}) = \mathbf{v}^T. \tag{2.4}$$

Normalize $\hat{\mathbf{x}}^T$ as $\hat{\mathbf{x}}^T/\|\hat{\mathbf{x}}^T\|_1$, where $\|\hat{\mathbf{x}}^T\|_1$ is the L_1-norm of $\hat{\mathbf{x}}^T$. Then the normalized $\hat{\mathbf{x}}^T$ is the PageRank vector.

2.6 Two-Way Reordered PageRank Model

Hsu et al noticed the existence of reverse-dangling nodes [9]. They categorize the reverse-dangling nodes into one class, and the dangling nodes into another class. Thus, the vertices of $\mathbf{G}$ are split into three disjoint classes V_{RD}, V_N, and V_D such that $V = V_{RD} \cup V_N \cup V_D$. The classes V_{RD}, V_N, and V_D are, respectively, defined as follow:

$$V_{RD} = \{v \in V | in(v) = 0\},$$

$$V_N = \{v \in V | in(v) > 0 \text{ and } out(v) > 0\}, \text{ and}$$

$$V_D = \{v \in V | out(v) = 0 \text{ and } in(v) > 0\},$$

where $in(v)$ and $out(v)$ respectively denote the in-degree and out-degree of node v. The set V_{RD} contains all reverse-dangling nodes including isolated nodes, V_D contains all dangling nodes, and V_N contains all normal nodes. For convenience, we call reverse-dangling node as RD-node, dangling node as D-node, and normal node as N-node.

Hsu et al reorder the indices of Web pages such that the RD-nodes and D-nodes, are respectively arranged in the front and rear of the Web pages list. The resulting hyperlink matrix $\mathbf{H}$

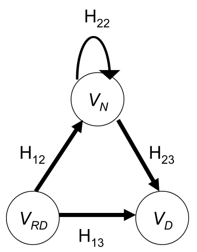

Figure 2.1: The transition relationship of sets V_{RD}, V_N, and V_D in the Web graph.

should become:

$$\mathbf{H} = \begin{array}{c} \\ V_{RD} \\ V_N \\ V_D \end{array} \begin{array}{c} \begin{array}{ccc} V_{RD} & V_N & V_D \end{array} \\ \begin{pmatrix} 0 & \mathbf{H}_{12} & \mathbf{H}_{13} \\ 0 & \mathbf{H}_{22} & \mathbf{H}_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{2.5}$$

We can depict the transition relationship of sets V_{RD}, V_N, and V_D as Fig. 2.1. In [9], Hsu et al prove the following theorem which splits the computation of PageRank vector π^T as three parts: RD-nodes, N-nodes, and D-nodes.

Theorem 2. ([9]) After reordering the pages, the hyperlink matrix $\mathbf{H}$ of Web would be as Eq. (2.5). Partition the personalization vector $\mathbf{v}^T$ into $\left(\mathbf{v}_1^T, \mathbf{v}_2^T, \mathbf{v}_3^T\right)$ according to reverse-dangling $\mathbf{v}_1^T$, normal $\mathbf{v}_2^T$, and dangling $\mathbf{v}_3^T$. Then, the unnormalized PageRank vector can be written as

$$\hat{\mathbf{x}}^T = \left(\hat{\mathbf{x}}_1^T, \hat{\mathbf{x}}_2^T, \hat{\mathbf{x}}_3^T\right), \tag{2.6}$$

where

$$
\begin{aligned}
\hat{\mathbf{x}}_1^T =& \mathbf{v}_1^T, \\
\hat{\mathbf{x}}_2^T =& \alpha \mathbf{v}_1^T \mathbf{H_{12}} \left(\mathbf{I} - \alpha \mathbf{H_{22}}\right)^{-1} + \mathbf{v}_2^T \left(\mathbf{I} - \alpha \mathbf{H_{22}}\right)^{-1}, \\
\hat{\mathbf{x}}_3^T =& \alpha \mathbf{v}_1^T \left(\mathbf{H_{13}} + \alpha \mathbf{H_{12}} \left(\mathbf{I} - \alpha \mathbf{H_{22}}\right)^{-1} \mathbf{H_{23}}\right) + \\
& \alpha \mathbf{v}_2^T \left(\mathbf{I} - \alpha \mathbf{H_{22}}\right)^{-1} \mathbf{H_{23}} + \mathbf{v}_3^T. \qquad (2.7)
\end{aligned}
$$

As shown, the unnormalized PageRanks of RD-nodes (i.e. $\hat{\mathbf{x}}_1^T$) are completely determined by their corresponding personalization values. For N-nodes (i.e. $\hat{\mathbf{x}}_2^T$), the PageRank problem can be regarded as the following linear system:

$$\hat{\mathbf{x}}_2^T (\mathbf{I} - \alpha \mathbf{H}_{22}) = \mathbf{v}_2^T + \alpha \mathbf{v}_1^T \mathbf{H}_{12}. \qquad (2.8)$$

To solve this linear system, we can use iterative method like Jacobi method ([4], page 8) or Gauss-Seidel method ([4], page 9) to solve it. With forward substitution, the PageRank problem for D-nodes (i.e. $\hat{\mathbf{x}}_3^T$) can be written as the formulation:

$$\hat{\mathbf{x}}_3^T = \alpha(\hat{\mathbf{x}}_1^T \mathbf{H}_{13} + \hat{\mathbf{x}}_2^T \mathbf{H}_{23}) + \mathbf{v}_3^T. \qquad (2.9)$$

Thus, with forward substitution, the computation for D-nodes is straight (i.e. no iteration is needed). Since the unnormalized PageRanks of RD-nodes are exactly their corresponding personalization values, the time of computing unnormalized PageRanks of RD-nodes is zero. With forward substitution, the computing cost for D-nodes is proportional to the number of in-links of

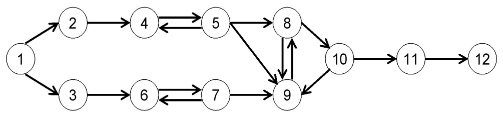

Figure 2.2: A tiny web contains 12 pages.

all D-nodes. If we use Jacobi's method to solve Eq. (2.8), we need $O\left(\frac{\log(\tau)}{\log(\alpha)} nnz(\mathbf{H}_{22})\right)$ operations to reach a convergence tolerance of τ, where $nnz(\mathbf{H}_{22})$ denotes the number of nonzeros in matrix $\mathbf{H}_{22}$ [14]. Comparing to Page et al's original PageRank model, Hsu et al efficiently reduce the size of the linear system.

Since such reordering efficiently reduces the size of the linear system, we can repeat the reordering to find some more "sub-reverse-dangling" and "sub-dangling" nodes. Consider the hyperlink matrix in Eq. (2.5). If we remove the RD-nodes, D-nodes, and their link arcs, we can recursively find some more zero-columns or zero-rows in the remaining sub-matrix. This means that we can get some more RD-nodes and D-nodes. Thus, recursively doing this reordering, the number of N-nodes becomes even smaller and the size of the linear system becomes even smaller. To be explicit, we define the following terms.

Definition 1. Assume that we do the reordering r times for retrieving RD-nodes, and d times for retrieving D-nodes. The RD-node found in i^{th} round reordering is defined as i-level RD-node, and the D-node found in $(d-i+1)^{th}$ round reordering is defined as i-level D-node.

Using the tiny web in Fig. 2.2 as an example, node 1 is 1-level RD-node, nodes 2 and 3 are 2-level RD-nodes, node 11 is 1-level D-node, and node 12 is 2-level D-node. After the recursively

$$\mathbf{H} = \begin{pmatrix} \mathbf{0} & \mathbf{H}_{12} & \mathbf{H}_{13} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{H}_{1(r+d+1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_{23} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{H}_{2(r+d+1)} \\ \vdots & & \ddots & \ddots & & & & & & & \vdots \\ \vdots & & & \ddots & \ddots & & & & & & \vdots \\ \vdots & & & & \mathbf{0} & \ddots & & & & & \vdots \\ \vdots & & & & & \mathbf{H}_{(r+1)(r+1)} & \ddots & & & & \vdots \\ \vdots & & & & & & \mathbf{0} & \ddots & & & \vdots \\ \vdots & & & & & & & \ddots & \ddots & & \vdots \\ \vdots & & & & & & & & \ddots & \ddots & \vdots \\ \mathbf{0} & & & & & & & & & \mathbf{0} & \mathbf{H}_{(r+d)(r+d+1)} \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{2.10}$$

reordering, the final matrix should be permuted as Eq. (2.10), where r is the number of blocks of RD-nodes, and d is the number of blocks of D-nodes. In Eq. (2.10), $\mathbf{H}_{(r+1)(r+1)}$ denotes the hyperlink matrix for N-nodes. Hsu et al respectively permute the i-level RD-nodes in front of $(i+1)$-level RD-nodes, and the i-level D-nodes in front of $(i+1)$-level D-nodes. Thus, we can continuously reduce the size of the linear system, and the computational complexity can be further reduced [9]. The complete two-way reordered PageRank algorithm is listed in Algorithm 1. In implementation, we can restrict the number of total blocks of RD-nodes and D-nodes to accelerate the reordering. Under this situation, though there may be some more sub-reverse-dangling and sub-dangling nodes, we regard them as N-nodes.

Algorithm 1 Two-Way Reordered PageRank

```
Input: H, α, v
Output: π (PageRank vector)
1: Reordering
   Use two-way reordering method to permute the indices of Web pages. Recursively do this
   reordering $r$ times for retrieving $RD$-nodes, and $d$ times for retrieving $D$-nodes. Thus, for
   $i = 1$ to $r$, $\mathbf{H}_{ii}$ represents the hyperlink matrix of $i$-level $RD$-nodes, for $i = (r+1)$, $\mathbf{H}_{ii}$
   represents the hyperlink matrix of $N$-nodes, and for $i = (r+2)$ to $(r+d+1)$, $\mathbf{H}_{ii}$ represents
   the hyperlink matrix of $(i-r-1)$-level $D$-nodes.
2: To solve the linear system
   2.1 $\mathbf{x}_1 = \mathbf{v}_1$.
   2.2 For $i = 2$ to $r$, compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji} + \mathbf{v}_i^T$.
   2.3 For $i = (r+1)$, solve for $\mathbf{x}_i^T$ in $\mathbf{x}_i^T(\mathbf{I} - \alpha\mathbf{H}_{ii}) = \mathbf{x}_i^{*T} + \mathbf{v}_i^T$, where $\mathbf{x}_i^{*T} = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji}$.
   2.4 For $i = (r+2)$ to $(r+d+1)$, compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji} + \mathbf{v}_i^T$.
3: Normalization
   $\pi^T = \left(\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_{r+d+1}^T\right) / \| \left(\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_{r+d+1}^T\right) \|_1$.
```

Chapter 3

SCC PageRank Algorithm

In this section, we discuss the details of SCC PageRank algorithm and how it addresses the personalized PageRank probelm. The SCC PageRank is developed on top of the two-way reordered PageRank. We combine the largest strongly connected component structure in Web graph with the two-way reordered PageRank. The space and time complexities of SCC PageRank are analyzed in Chapter 3.4.

3.1 Two-Way Reordered PageRank on Personalized PageRank

In two-way reordered PageRank, there is a special phenomenon. The unnormalized PageRanks of 1-level RD-nodes are completely determined by their corresponding personalization values (see step 2.1 in Algorithm 1). For 2-level RD-node, its unnormalized PageRank is its corresponding personalization value plus the sum of the unnormalized PageRanks of all 1-level RD-nodes pointing into this node (see step 2.2 in Algorithm 1 and recall Eq. (2.1)). For 3-level RD-node, its unnormalized PageRank is similar except that the nodes, pointing into this node, also include

2-level RD-nodes. Inductively, all unnormalized PageRanks of RD-nodes depend on their ancestors. Totally, the computing cost for all RD-nodes is propotional to the number of all their in-links (i.e. the computing cost is $O(in(RD\text{-nodes}))$, where $in(RD\text{-nodes})$ is the number of total in-links of all RD-nodes). For D-nodes, the situation is similar. 1-level D-nodes accumulate the unnormalized in-PageRanks from RD-nodes and N-nodes, and then give their unnormalized PageRanks to all their out-neighbors, and so on. Totally, the computing cost for RD-nodes and D-nodes is $O(in(RD\text{-nodes}+D\text{-nodes}))$. If we exclude the N-nodes, the computation for RD-nodes and D-nodes can be regarded as a propagation. In this propagation, we compute the unnormalized PageRanks level by level. While computing new personalized PageRank vector π^*, if the personalization values of nodes, which are at preceding levels of the propagation, stay unchanged, we can directly get their unnormalized PageRanks from π, and thus get an improvement in performance.

This idea sounds feasible. However, there is a problem caused by N-nodes. All N-nodes form a single block in two-way reordered PageRank. We use the iterative method like Jacobi to solve the corresponding linear system. When using Jacobi to solve the linear system of N-nodes, $O\left(\frac{\log(\tau)}{\log(\alpha)} nnz(\mathbf{H}_{(r+1)(r+1)})\right)$ operations are needed [14]. The parameter τ is convergence tolerance which is used to detect whether the computation of stationary vector completes. In [8], the parameter τ is set to 10^{-10} and the parameter α is set to 0.9. Thus the computation needs to scan the hyperlink matrix of N-nodes, $\mathbf{H}_{(r+1)(r+1)}$, about 218 times. Comparing this time with the time of computing RD-nodes and D-nodes, the computation for N-nodes consumes much more time. The improvement in computing RD-nodes and D-nodes is diluted. Thus, we have to improve the computation for N-nodes.

3.2 SCC PageRank

To improve the computation for N-nodes, we use Example 1 to analyze the block of N-nodes.

Example 1. Consider the tiny web in Fig. 2.2. Nodes 4 - 10 are N-nodes. The linear system for these nodes is listed below:

1. $\mathbf{x}_4 - 1/3\alpha\mathbf{x}_5 = \mathbf{v}_4 + \alpha\mathbf{x}_2$,
2. $\mathbf{x}_5 - \alpha\mathbf{x}_4 = \mathbf{v}_5$,
3. $\mathbf{x}_6 - 1/2\alpha\mathbf{x}_7 = \mathbf{v}_6 + \alpha\mathbf{x}_3$,
4. $\mathbf{x}_7 - \alpha\mathbf{x}_6 = \mathbf{v}_7$,
5. $\mathbf{x}_8 - \alpha\mathbf{x}_9 = \mathbf{v}_8 + 1/3\alpha\mathbf{x}_5$,
6. $\mathbf{x}_9 - 1/2\alpha\mathbf{x}_8 - 1/2\alpha\mathbf{x}_{10} = \mathbf{v}_9 + 1/3\alpha\mathbf{x}_5 + 1/2\alpha\mathbf{x}_7$,
7. $\mathbf{x}_{10} - 1/2\alpha\mathbf{x}_8 = \mathbf{v}_{10}$,

where $\mathbf{x}_i$ represents the unnormalized PageRank of node i. From this linear system, one can see that if we solve equations 1 and 2 simultaneously, we only need the solution of $\mathbf{x}_2$. Similarly, solving $\mathbf{x}_6$ and $\mathbf{x}_7$ (equations 3 and 4) only need the solution of $\mathbf{x}_3$. After solving these four

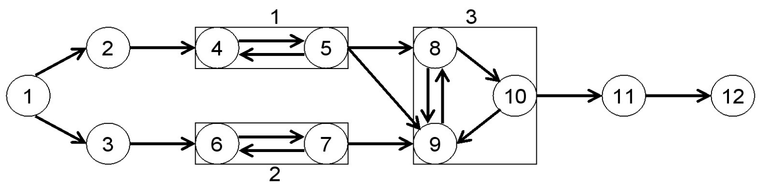

Figure 3.1: SCC Decomposition for N-nodes in the tiny web in Fig. 2.2.

equations, we can solve equations 5, 6, and 7. This computation is similar to the propagated computation for RD-nodes and D-nodes. In Fig. 2.2, if the personalization values of nodes 1, 2, 4, and 5 always stay unchanged, we do not need to recompute their unnormalized PageRanks while computing new personalized PageRank vector π^*. $\square$

From Example 1, one can see that if node i is the ancestor of node j, we have to solve node i before solving node j. If a pair of nodes are both ancestors of each other, we have to solve their corresponding equations simultaneously. In graph theory, if a pair of nodes are both ancestors of each other, they belong to the same strongly connected component (SCC). So we decompose the N-nodes subgraph into largest SCCs and each SCC represents a computational unit. Using Fig. 2.2 as an example, its SCC decomposition for N-nodes is depicted in Fig. 3.1.

After finding the computational units, we have to decide the computing order for each computational unit. To do this, we firstly construct the corresponding SCC graph ([16], page 226) of SCCs of N-nodes. The nodes of the SCC graph correspond to the SCCs. There is a directed edge from SCC$_i$ to SCC$_j$ if any node in SCC$_i$ have links to any node in SCC$_j$. Using Fig. 3.1 as an example, its SCC graph is depicted in Fig. 3.2. In ([16], page 226), it indicates that the SCC graph is acyclic. Thus, to decide the order of computation, we can regard the SCC graph as an

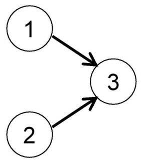

Figure 3.2: The corresponding SCC graph of SCCs of N-nodes in Fig. 3.1

Activity On Vertex Network (AOV-Network) and use topological sorting to sort the SCC graph. The resulting sequence is the computing order sequence.

The topological sorting outputs several SCCs at each round. So we can use levels to index the SCCs. An SCC outputted at the first round is called 1-level SCC, an SCC outputted at the second round is called 2-level SCC, and so on. Using SCC graph in Fig. 3.2 as an instance, nodes 1 and 2 are 1-level SCCs, and node 3 is 2-level SCC. Now we combine the SCC structures with two-way reordered PageRank. Assuming there are total n levels of N-nodes, the hyperlink matrix for N-nodes in Eq. (2.10) can be expanded to:

$$\mathbf{H}_{(r+1)(r+1)} = \begin{pmatrix} \mathbf{A}_{11} & \cdots & \cdots & \mathbf{A}_{1n} \\ \mathbf{0} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_{nn} \end{pmatrix}. \tag{3.1}$$

Thus, the whole propagated computation is constructed. We can depict the propagated computation as Fig. 3.3. Using this propagated computation, the unnormalized PageRanks are computed level by level. We call the whole processes as SCC PageRank and list it in Algorithm 2.

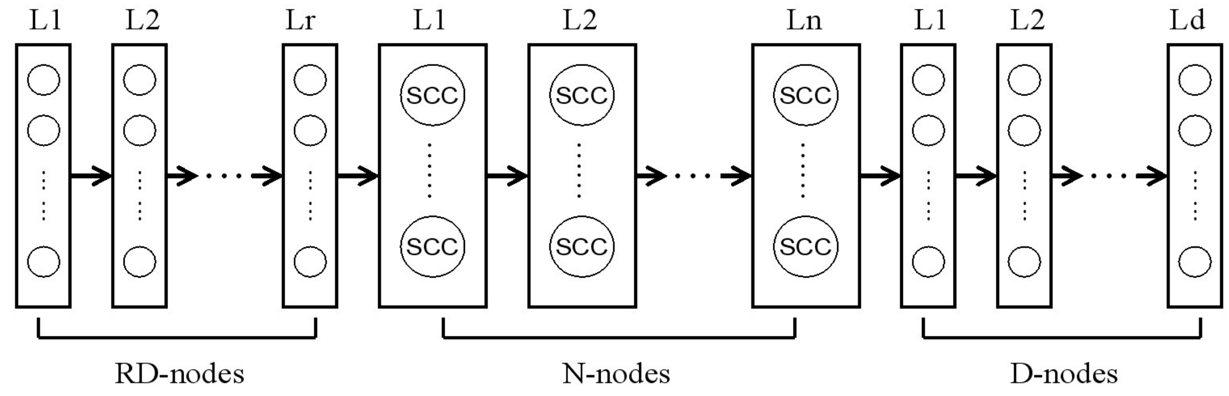

Figure 3.3: The propagated computation for the PageRank problem. Assume there are total r levels of RD-nodes, total n levels of N-nodes, and total d levels of D nodes. The label, $\mathbf{L}i$, of each kind of nodes represents its level.

Algorithm 2 SCC PageRank

```
Input: H, $\alpha$, v
Output: $\pi$ (PageRank vector)
1: Reordering.
   Use two-way reordering method to permute the indices of Web pages. Recursively doing this reordering $r$ times for retriving $RD$-nodes, and $d$ times for retriving $D$-nodes.
2: SCC Decomposition.
   Do SCC decomposition for $N$-nodes.
3: SCC graph Construction
   Construct the SCC graph for $N$-nodes subgraph.
4: Topological Sorting.
   Topologically sort the SCC graph. Assume the topological sorting runs $n$ rounds.
5: To solve the linear system.
   5.1 $\mathbf{x}_1 = \mathbf{v}_1$.
   5.2 For $i = 2$ to $r$, compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji} + \mathbf{v}_i^T$.
   5.3 For $i = (r+1)$ to $(r+n)$
       For $k = 1$ to $w$, where $w$ is the number of SCCs at this level
           solve for $\mathbf{y}_k^T$ in $\mathbf{y}_k^T(\mathbf{I} - \alpha \mathbf{H}_{i_k i_k}) = \mathbf{y}_k^{*T} + \mathbf{v}_{i_k}^T$, where $\mathbf{y}_k^{*T} = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{j i_k}$.
       $\mathbf{x}_i^T = (\mathbf{y}_1^T, \mathbf{y}_2^T, \cdots, \mathbf{y}_w^T)$.
   5.4 For $i = (r+n+1)$ to $(r+n+d)$, compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji} + \mathbf{v}_i^T$.
6: Normalization.
   $\pi^T = \left(\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_{r+n+d}^T\right) / \| \left(\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_{r+n+d}^T\right) \|_1$.
```

3.3 SCC PageRank on Personalized PageRank

The main spirit of SCC PageRank is exploiting propagated computation. In computing the new personalized PageRank vector, π^*, if the personalization values of nodes, which are at the preceding levels of the propagation, stay unchanged, we do not need to recompute their unnormalized PageRanks. Thus, we can get an improvement in performance. Since the individual personalization value may be mutable (recall that the personalization value is a kind of probability), we use a specific implementation of personalization vector. We assign each page a rating score, which denotes a surfer's interest in this page. Each rating score is a positive number. We set a vector $\mathbf{u}$ to hold all the rating scores of all Web pages. With this rating vector, $\mathbf{u}$, the personalization vector, $\mathbf{v}$, can be computed as follow:

$$\mathbf{v} = \mathbf{u}/\|\mathbf{u}\|_1. \tag{3.2}$$

Thus, the personalization vector is still positive. In a uniformly distributed personalization vector, each page has equal rating score and thus has equal personalization value (i.e. $1/n$). With this implementation, we discuss how SCC PageRank works on computing new personalized PageRank vector, π^*.

Given the old rating vector is $\mathbf{u}$, the new rating vector is $\mathbf{u}^*$ and the constant $c = \frac{\|\mathbf{u}\|_1}{\|\mathbf{u}^*\|_1}$, the following theorem indicates the new unnormalized PageRanks of all kinds of nodes.

Theorem 3. The new unnormalized PageRanks are:

1. For each RD-node, who has unchanged rating score, if it has no ancestors, then its new unnormalized PageRank is c times its old one. If it has ancestors, and all its ancestors have unchanged rating scores, then its new unnormalized PageRank is c times its old one.

2. For each SCC, $\mathbf{K}$, in N-nodes, if $\mathbf{K}$ has no ancestors, and $\mathbf{u}^*_{\mathbf{K}} = \mathbf{u}_{\mathbf{K}}$, then the new unnormalized PageRanks of $\mathbf{K}$ are c times their old ones. If $\mathbf{K}$ have ancestors, and $\mathbf{u}^*_{\mathbf{K}} = \mathbf{u}_{\mathbf{K}}$, then the new unnormalized PageRanks of $\mathbf{K}$ are c times their old ones as long as all ancestors of $\mathbf{K}$ have unchanged rating scores.

3. For each D-node, if all its ancestors have unchanged rating scores, then its new unnormalized PageRank is c times its old one.

4. In other situation, we have to compute the new unnormalized PageRanks for the nodes.

To prove Theorem 3, we firstly prove the follwing lemmas.

Lemma 1. Given that the solution of the linear system $\mathbf{Ax} = \mathbf{b}$ is $\hat{\mathrm{x}}$, then the solution of the linear system $\mathbf{Ax} = \mathbf{cb}$ is $\mathbf{c}\hat{\mathbf{x}}$, where $\mathbf{c}$ is a constant.

Proof. The known condition, $\hat{\mathrm{x}}$ is the solution of $\mathbf{Ax} = \mathbf{b}$, implies $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$. By multiplying $\mathbf{c}$ to both sides of this equation, we can get $\mathbf{cA}\hat{\mathbf{x}} = \mathbf{cb} \Rightarrow \mathbf{A}(\mathbf{c}\hat{\mathbf{x}}) = \mathbf{cb}$. Thus, $\mathbf{c}\hat{\mathbf{x}}$ is the solution of the linear system $\mathbf{Ax} = \mathbf{cb}$. □

Lemma 2. The new personalization value of node i, $\mathbf{v}^*_i$, is $\frac{\|\mathbf{u}\|_1}{\|\mathbf{u}^*\|_1}$ times its old one as long as $\mathbf{u}_i = \mathbf{u}^*_i$.

Proof. By Eq. (3.2), the old and new personalization values of a node are $\mathbf{v}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}\|_1}$ and $\mathbf{v}_i^* = \frac{\mathbf{u}_i^*}{\|\mathbf{u}^*\|_1}$ respectively. This implies $\mathbf{v}_i\|\mathbf{u}\|_1 = \mathbf{u}_i$ and $\mathbf{v}_i^*\|\mathbf{u}^*\|_1 = \mathbf{u}_i^*$. Since $\mathbf{u}_i = \mathbf{u}_i^*$, $\mathbf{v}_i\|\mathbf{u}\|_1 = \mathbf{v}_i^*\|\mathbf{u}^*\|_1$. Thus, $\mathbf{v}_i^* = \mathbf{v}_i\frac{\|\mathbf{u}\|_1}{\|\mathbf{u}^*\|_1}$. □

Now we prove the Theorem 3.

Proof. The sufficient conditions are:

1. The new rating score of the node is the same as its old one.

2. All ancestors of the node have unchanged rating scores.

We split the proof into three parts: RD-nodes, N-nodes, and D-nodes.

1. RD-**nodes:**

   If a node i is RD-node and has no ancestors, then it must be 1-level RD-node. We have indicated that the unnormalized PageRank of 1-level RD-node is its corresponding personalization value. Thus, the new and old unnormalized PageRanks of node i are $\mathbf{v}_i^*$ and $\mathbf{v}_i$ respectively. By sufficient condition 1 and lemma 2, we can get $\mathbf{v}_i^* = c\mathbf{v}_i$. Thus, the new unnormalized PageRank of node i is c times its old one. For 2-level RD-nodes, by step 5.2 in Algorithm 2, we know that the old unnormalized PageRanks of these nodes are $\mathbf{x}_2^T = \alpha\mathbf{x}_1^T\mathbf{H}_{12} + \mathbf{v}_2^T$, and their new unnormalized PageRanks are $\mathbf{x}_2^{*T} = \alpha\mathbf{x}_1^{*T}\mathbf{H}_{12} + \mathbf{v}_2^{*T}$. The ancestors of 2-level RD-nodes must be 1-level RD-nodes. We have proved that $\mathbf{x}_1^* = c\mathbf{x}_1$. By sufficient condition 1 and lemma 2, we can get $\mathbf{v}_2^* = c\mathbf{v}_2$. Combining these 2 results, the right hand side of the equation is c times its old one. Thus, the new unnormalized PageRanks of 2-level RD-nodes are c times their old ones. For other RD-nodes, the proof is similar.

2. N**-nodes:**

   For N-nodes, we have to consider for each SCC. By step 5.3 in Algorithm 2, the unnormalized PageRanks of an SCC is the solution $\mathbf{y}_k^T$ in the linear system $\mathbf{y}_k^T(\mathbf{I} - \alpha \mathbf{H}_{i_k i_k}) = \mathbf{y}_k^{*T} + \mathbf{v}_{i_k}^T$, where $\mathbf{y}_k^{*T} = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{j i_k}$. One can see that the right hand side of the linear system is the unnormalized PageRanks of the ancestors plus the corresponding personalization values. By sufficient condition 1 and lemma 2, we can get $\mathbf{v}_{i_k}^{*T} = c\mathbf{v}_{i_k}^T$. Thus, if 1-level N-nodes have no ancestors, by lemma 1, we can prove the theorem. If 1-level N-nodes have ancestors, the ancestors must be RD-nodes. By sufficient condition 2 and proof for RD-nodes, we can get the new unnormalized PageRanks of the ancestors are c times their old ones. Thus, the right hand side of the linear system is c times its old one. By lemma 1, we can prove the theorem. For 2-level N-nodes, their ancestors must be RD-nodes or 1-level N-nodes. By the preceding proof and lemma 1, we can get the result that the right hand side of the linear system is c times its old one. Thus, we can prove the theorem for 2-level N-nodes. For other N-nodes, the proof is similar.

3. D**-nodes:**

   The ancestors of 1-level D-nodes must be RD-nodes or N-nodes. By sufficient condition

2 and preceding proofs, we can get that the new unnormalized PageRanks of the ancestors are c times their old ones. By sufficient condition 1 and lemma 2, the new personalization values of these D-nodes are c times their old ones. Combining these results, the right hand side of the equation, which is in step 5.4 in Algorithm 2, is c times its old one. Thus, we can prove the theorem for 1-level D-nodes. For other D-nodes, the proof is similar.

□

We can interpret Theorem 3 as: If all ancestors of a set of nodes have unchanged rating scores, and the rating scores of nodes in this set are also unchanged, then the new unnormalized PageRanks of these node are $\frac{\|\mathbf{u}\|_1}{\|\mathbf{u}^*\|_1}$ times their old unnormalized PageRanks. For N-nodes, if the sufficient condition is true, we do not need to iteratively compute their new unnormalized PageRanks. Instead, we can get their new unnormalized PageRanks by multiplying $\frac{\|\mathbf{u}\|_1}{\|\mathbf{u}^*\|_1}$ to their old ones. In other words, if the rating score of a node changes, we have to recompute for all its descendents. For a specific class of users, they have the same interest in most of Web pages. We can define a standard rating score to represent that the user have no specific interest in this page. Thus, the rating scores of most Web pages stay unchanged while computing personalized PageRank vectors. By this way, we expect that the improvement is feasible.

To exploit Theorem 3, we have to check the necessary recomputations. The checking algorithm is listed in Algorithm 3. The whole algorithm for computing personalized PageRank vectors includes running the checking algorithm, computing the unnormalized PageRanks for necessary parts, and normalizing the unnormalized PageRank vector.

Algorithm 3 Detecting Necessary Recomputation

```
Input: $\mathbf{H}$, $\mathbf{r}$, $\mathbf{r}^*$
Output: Necessary Recomputation (The marked parts)
1: Assume there are total $r$ levels of $RD$-nodes, total $n$ levels of $N$-nodes, and total $d$ levels of
   $D$-nodes.
2: For $i = 1$, (1-level $RD$-nodes)
     If($\mathbf{r}_i^* \neq \mathbf{r}_i$)
       mark the node, whose rating score changes, and all its out-neighbors.
3: For $i = 2$ to $r$, (other $RD$-nodes)
     If(any node at this level has been marked)
       mark all out-neighbors of the node, who is marked.
     If($\mathbf{r}_i^* \neq \mathbf{r}_i$)
       mark the node, whose rating score changes, and all its out-neighbors.
4: For $i = (r + 1)$ to $(r + n)$, ($N$-nodes)
     If(any SCC at this level has been marked)
       mark all out-neighbors of this SCC.
     If($\mathbf{r}_i^* \neq \mathbf{r}_i$)
       mark the SCC, whose nodes have changed rating score, and mark all out-neighbors of
   this SCC.
5: For $i = (r + n + 1)$ to $(r + n + k)$, ($D$-nodes)
     If(any node at this level has been marked)
       mark all out-neighbors of the node, who is marked.
     If($\mathbf{r}_i^* \neq \mathbf{r}_i$)
       mark the node, whose rating score changes, and all its out-neighbors.
```

3.4 Time and Space Complexities of SCC PageRank

In this section, we formally derive the time complexity and additional storage of SCC PageRank.

Running SCC PageRank for computing single PageRank vector needs the following steps:

1. **Reordering:** Let $\mathbf{B}$ be the predetermined maximum number of square diagonal blocks in the reordered hyperlink matrices. The time of reordering is $O(\mathbf{B} \times n(\mathbf{H}))$ [8].

2. **SCC Decomposition:** The SCC decomposition takes $O(|V| + |E|)$ time ([16], page 230). That is, $O(n(\mathbf{H}^*) + nnz(\mathbf{H}^*))$, where $\mathbf{H}^*$ is $\mathbf{H}_{(N\text{-nodes}) \times (N\text{-nodes})}$.

3. **Construction of SCC graph:** The construction of SCC graph needs to check the edges between all pairs of SCCs. Since these edges are exactly the links of N-nodes, the time complexity is $O(nnz(\mathbf{H}_{(N\text{-nodes})\times(N\text{-nodes})}))$.

4. **Topological Sorting:** The topological sorting takes $O(|V| + |E|)$ time ([16], page 203). That is, O(n(SCC graph) + nnz(SCC graph)).

5. **Computing Unnormalized PageRank:** Computing unnormalized PageRank includes 3 parts: RD-nodes, N-nodes, and D-nodes. We have indicated that the computations for RD-nodes and D-nodes takes $O(in(RD\text{-nodes} + D\text{-nodes}))$ time. The time of computation for N-nodes is the sum of time of computing each SCC. For each SCC, the Jacobi's iteration requires $O(\frac{log(\tau)}{log(\alpha)} nnz(\text{SCC}))$ operations [14]. Thus, the total time is: $O(\frac{log(\tau)}{log(\alpha)} \sum_{\text{all SCC}} nnz(\text{SCC}))$ $\leq O(\frac{log(\tau)}{log(\alpha)} nnz(\mathbf{H}_{(N\text{-nodes})\times(N\text{-nodes})}))$.

6. **Normalization:** The time of normalization is $O(n(\mathbf{H}))$.

In general, $n(\mathbf{SCCG}) \leq n(N\text{-nodes})$, $nnz(\mathbf{SCCG}) \leq nnz(\mathbf{H}_{(N\text{-nodes})\times(N\text{-nodes})})$. Thus, the time complexity of SCC PageRank can be formulated by the following theorem.

Theorem 4. Let B be the predetermined maximum number of square diagonal blocks in the reordered hyperlink matrix. If we use the Jacobi's method to solve the small linear subsystem of each SCC, the time complexity of SCC PageRank is $O(\mathbf{B} \times n(\mathbf{H}) + (\frac{log(\tau)}{log(\alpha)} + c) nnz(\mathbf{H}^*))$, where c is a constant, τ is the convergence tolerance of Jacobi's method, α is the damping factor, and

$\mathbf{H}^*$ is $\mathbf{H}_{(N\text{-nodes})\times(N\text{-nodes})}$.

When using SCC PageRank to compute different personalized PageRank vectors, we need detecting the change in rating vector, computing the unnormalized PageRank for necessary parts, and normalizing the unnormalized PageRank vector. The time for detecting change is $O(n(\mathbf{H}) + nnz(\mathbf{H}))$. Thus, when using SCC PageRank to compute the different personalized PageRank vectors, the time is $O(n(\mathbf{H}) + nnz(\mathbf{H}))$ plus the time of computing unnormalized PageRank vector and normalization.

The additional storage for SCC PageRank includes following items.

1. **SCC Index Sequence:** This sequence is used to store the index of SCC for each node. Since each N-node belongs to only one SCC, the total storage of this sequence is $O(n(N\text{-nodes}))$.
2. **SCC graph:** The SCC graph is a graph. Using sparse matrix [18] to store this graph, the storage is $O(nnz(\mathbf{SCCG}))$.
3. **Computing Order Sequence:** This sequence is used to indicate the computing order of all SCCs. Thus, the storage is $O(n(\mathbf{SCCG}))$.

When using SCC PageRank on computing different personalized PageRank vectors, we need to store the rating vector, and it requires $O(n(\mathbf{H}))$ storage.

Chapter 4

Experimental Evaluation

To evaluate the performance of SCC PageRank, we do some experiments. We implement this algorithm in MATLAB. The experiments are conducted on a 2.11 GHz AMD Athlon 64 X2 Dual Core Processor with 2.0 GB RAM. Same as the experiments in [8] and [14], we set $\alpha = 0.9$ as the damping factor, $\tau = 10^{-10}$ as the convergence tolerance, and initial personalization vector is uniformly distributed (i.e. $\mathbf{v}^T = [1/n, 1/n, \ldots, 1/n]$). The maximum number of square diagonal blocks in the reordered matrices is limited to 12 (i.e. N-nodes has 1 block, RD-nodes and D-nodes have total 11 blocks). Also, we use Jacobi's method to solve the linear subsystems. We use four datasets as the input tiny webs. Dataset Hollins is generated by Webbot crawler, which contains 6013 nodes and 23875 links [1]. The California dataset contains 9664 nodes and 16150 links matching to the query topic of "California" [2]. Dataset WWW was used in [2] to predict an average distance of the Web based on their crawl of nd.edu site [3]. WWW is a distinctive subset of the Web, which contains 325729 nodes and 1497134 links. In WWW dataset, all nodes have

[1]Hollins was from http://www.limfinity.com/ir/

[2]California was from http://www.cs.cornell.edu/Courses/cs685/2002fa/

[3]WWW was from http://www.nd.edu/networks/resources.htm

at least one in-link. Thus there is no RD-node in this dataset. Stanford-Berkeley dataset was generated from a crawl of the stanford.edu and berkeley.edu domains created in December 2002 by the Stanford WebBase project [4].

Table 4.1: The structures of the datasets

Content	Hollins	California	WWW	Stanford-Berkeley
$n(\mathbf{H})$	6013	9664	325729	683446
$nnz(\mathbf{H})$	23875	16150	1497134	7583376
$n(RD$-nodes)	100	8003	0	88057
$n(N$-nodes)	2474	346	124887	588895
$n(D$-nodes)	3439	1345	200842	6494
$n(\mathbf{SCCG})$	91	89	2315	9104
$nnz(\mathbf{SCCG})$	95	84	3714	12189

Table 4.2: The execution time (in seconds) of the SCC PageRank, two-way reordered Pagerank, and power method

Algorithm	Step	Hollins	California	WWW	Stanford-Berkeley
SCC PageRank	Reordering	0.056471	0.059164	5.366798	12.164499
	SCC Decomposition	0.050321	0.003822	6.084947	32.577811
	SCCG Construction	0.004674	0.001449	2.056503	20.349997
	Topological Sort	0.004936	0.002076	0.018363	0.360674
	Computing PageRank	0.167345	0.047721	5.227131	56.740593
	Normalization	0.001075	0.000902	0.028297	0.097801
	Total	**0.284821**	**0.117135**	**18.782039**	**122.291375**
Two-Way Reordered PageRank	Reordering	0.056471	0.059164	5.366798	12.164499
	Computing PageRank	0.153114	0.036794	6.825858	60.180660
	Normalization	0.000502	0.000811	0.028091	0.098775
	Total	**0.210087**	**0.095358**	**12.220747**	**72.443934**
Power Method	Computing PageRank	0.235140	0.314254	15.252731	75.124361
	Normalization	0.000412	0.000628	0.027019	0.096550
	Total	**0.235552**	**0.314882**	**15.279750**	**75.220911**

The structures of these datasets are listed in Table 4.1, where $\mathbf{SCCG}$ is the abbreviation of SCC graph. The structures show that $n(\mathbf{SCCG}) + nnz(\mathbf{SCCG}) \ll nnz(\mathbf{H})$. Thus, the storage for SCC graph is small. For performance evaluation, we input the four datasets into the SCC

[4]Stanford-Berkeley was from http://www.kamvar.org/personalization

Table 4.3: The execution time of computing personalized PageRank vectors. SCC PageRank uses the worst case.

Algorithm	Step	Hollins	California	WWW	Stanford-Berkeley
SCC PageRank	Detecting Rating Vector	0.010134	0.018148	0.755640	3.734774
	Computing PageRank	0.167345	0.047721	5.227131	56.740593
	Normalization	0.001075	0.000902	0.028297	0.097801
	Total	**0.178554**	**0.066771**	**6.011068**	**60.573168**
Two-way Reordered	Computing PageRank	0.153114	0.036794	6.825858	60.180660
	Normalization	0.000502	0.000811	0.028091	0.098775
	Total	**0.153616**	**0.037605**	**6.853949**	**60.279435**
Power Method	Computing PageRank	0.235140	0.314254	15.252731	75.124361
	Normalization	0.000412	0.000628	0.027019	0.096550
	Total	**0.235552**	**0.314882**	**15.279750**	**75.220911**

Table 4.4: Number of nodes (NN) at each topological level and its corresponding computing time (in seconds).

Topological Level	Hollins		California		WWW		Stanford-Berkeley	
	NN	Time	NN	Time	NN	Time	NN	Time
1	86	0.0047	171	0.0238	53968	1.8151	76821	4.8420
2	6	0.0000	28	0.0005	27799	0.8432	37098	1.8888
3	1431	0.0593	8	0.0008	31919	1.9535	12723	0.7097
4	175	0.0356	17	0.0003	7581	0.2787	2826	0.1384
5	67	0.0104	29	0.0023	1181	0.0517	453	0.0236
6	701	0.0526	22	0.0013	2379	0.0656	2467	0.3477
7	8	0.0012	31	0.0106	38	0.0084	3348	0.1642
8			8	0.0010	11	0.0022	35	0.0049
9			2	0.0007	3	0.0004	40	0.0014
10					1	0.0003	45	0.0012
11 - 50					7	0.0014	452970	47.3053
51 - 95							69	0.0000

PageRank, two-way reordered PageRank, and power method. The power method is the simplest and most intuitive algorithm for computing PageRank. It has no extra processes and only iteratively compute the PageRank until the stationary PageRank is achieved. The experimental results are listed in Table 4.2. From Table 4.2, the experimental result shows that the computation of PageRank in SCC PageRank is fastest if the number of nodes in Web is large. In [3], Arasu et al used the SCC decomposition for accelerating the computation of PageRank. Our experimental result conforms their expectation. However, if the number of nodes in Web is small, the computation of PageRank in SCC PageRank is slower. This is because we have to do some I/O operations for retrieving the small linear subsystems. Thus, the I/O overhead lowers the performance. The experimental result also shows that the total running time of SCC PageRank is longer than that of the other two methods. The extra time is mainly consumed in SCC decomposition and construction of SCC graph. Since we do a DFS search in the SCC decomposition and a topological sorting, the I/O overhead results in the extra time penalty.

In computing different personalized PageRank vectors, we do not need to reconstruct the hierarchical structure of Web. We only need to detect the change in rating vector and recompute the necessary recomputation. The acceleration in computing different personalized PageRank vectors is hard to measure. Thus, we do an experiment to discuss the worst case. The worst case of SCC PageRank is that all Web pages need to be recomputed. Thus, in the worst case, SCC PageRank needs to check the change in rating vector and recompute PageRank for all nodes. The experimental result is listed in Table. 4.3. For WWW and Stanford-Berkeley datasets, the performance of worst case of SCC PageRank is about the same as that of two-way reordered PageRank, and is betterr than that of power method. If there is any saving of redundant recomputation, the SCC PageRank would perform better than two-way reordered PageRank. For Hollins and California datasets, the performance of SCC PageRank is worse than that of two-way reordered PageRank. This is resulted by the I/O overhead.

To discuss the possible improvement, we analyze the structures of N-nodes in four datasets. Since the topological order directly affects the performance of SCC PageRank, we compute the number of nodes at each topological level and their corresponding computing time. The results are listed in Table 4.4. Recall that indicated in Theorem. 3, if the rating score of a node changes, we have to recompute all its descendents. Thus, when checking the necessity of recomputation for N-nodes, we need to take the change of RD-nodes into account. For WWW dataset, the situation is special because it has no RD-nodes. The number of nodes, at first topological level in WWW, is 53968. In our statistic, these 53968 nodes belong to two different SCCs. If all these 53968 pages have unchanged rating scores, we can save about 1.8151 seconds at each recomputation. If surfers have different interest in any of these two SCCs, we have to recompute all its descendents. For Stanford-Berkeley dataset, the number of all RD-nodes plus the number of 1-level N-nodes is 164878. It is about 25% of the total number of pages in this dataset. If surfers only have different interest in other 75% pages, we can prevent much recomputation. How much redundant recomputation can be prevented depends on the hierarchical structure of Web, content of pages, and surfers' interest.

Chapter 5

Related Work

In this section, we review some important works related to this thesis. In [6], Broder et al propose the SCC structure in Web. In their research, the Web forms a structure which Broder et al call "bow-tie." The bow-tie structure is similar to the hierarchical structure in this thesis.

As mentioned in Chapter 1, there are much research elaborated on the personalized PageRank problem. We have mentioned two promising methods, the scalable personalized PageRank method and BlockRank. Since the spirit of both BlockRank and SCC PageRank is preventing some redundant recomputations, we indicate the differences between them as follow.

1. SCC PageRank can be looked as a top-down decomposition method. BlockRank can be looked as a button-up construction method.

2. In implementation, SCC PageRank needs a DFS search, and thus it takes more I/O operations than BlockRank needs.

3. The personalization vector used in BlockRank represents probabilities that surfer teleports between hosts. The personalization vector used in SCC PageRank represents probabilities

that surfer teleports between pages.

4. In computing personalized PageRank vectors, no matter what the difference between new personalization vector and old personalization vector is, BlockRank needs to run the standard PageRank algorithm once. However, SCC PageRank computes only some parts of the whole linear system under some circumstances.

Chapter 6

Conclusions

SCC PageRank algorithm decomposes the Web into hierarchical structure and uses this structure to compute the personalized PageRank vector. For computation of different personalized PageRank vectors, SCC PageRank avoids redundant recomputation and thus gets a performance improvement. By using the implementation of personalization vector we conducted, the improvement of SCC PageRank on personalized PageRank problem is feasible. The future work of the SCC PageRank includes: SCC link structure between crawled pages and uncrawled pages, choosing best base personalization vector such that we can get the least recomputation, utilization on spamming and other topics of personalized PageRank.

Bibliography

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison Wesley, 1983.

[2] Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[3] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms. In *Proceedings of the 11th International World Wide Web Conference, Poster Track*, 2002.

[4] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[5] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Pagerank as a function of the damping factor. In *Proceedings of the 14th international conference on World Wide Web*, pages 557–566, New York, NY, USA, 2005. ACM Press.

[6] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320, 2000.

[7] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 200–209, 2000.

[8] C. M. Hsu. *On the Processing and Measurement of High Dimensional Data*. PhD thesis.

[9] C. M. Hsu and M. S. Chen. Efficient web matrix processing based on dual reordering. In *Proceedings of ACM 17th Conference on Information and Knowledge Management*, 2008.

[10] G. Jeh and J. Widom. Scaling personalized web search. Technical report, Stanford University, 2002.

[11] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Exploiting the block structure of the web for computing pagerank. Technical Report 2003-17, Stanford University, 2003.

[12] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–400, 2004.

[13] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[14] A. N. Langville and C. D. Meyer. A reordering for the pagerank problem. *SIAM J. Scientific and Statistical Computing*, 27(6):2112–2120, 2006.

[15] C. P. Lee, G. H. Golub, and S. A. Zenios. A fast two-stage algorithm for computing pagerank. Technical Report SCCM-2003-15, Stanford University, 2003.

[16] Udi Manber. *Introduction to Algorithms: A Creative Approach*. Addison Wesleyd, 2003.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 2003-17, Stanford University, 1998.

[18] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[19] E. Seneta. *Non-negative Matrices and Markov Chains (Revised Printing)*. Springer, 2006.