Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 507915, 6 pages
http://dx.doi.org/10.1155/2013/507915

Research Article

An Adaptive Reordered Method for Computing PageRank

Yi-Ming Bu and Ting-Zhu Huang

School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

Correspondence should be addressed to Ting-Zhu Huang; tingzhuhuang@126.com

Received 8 May 2013; Accepted 7 July 2013

Academic Editor: Jinyun Yuan

Copyright © 2013 Y.-M. Bu and T.-Z. Huang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose an adaptive reordered method to deal with the PageRank problem. It has been shown that one can reorder the hyperlink matrix of PageRank problem to calculate a reduced system and get the full PageRank vector through forward substitutions. This method can provide a speedup for calculating the PageRank vector. We observe that in the existing reordered method, the cost of the recursively reordering procedure could offset the computational reduction brought by minimizing the dimension of linear system. With this observation, we introduce an adaptive reordered method to accelerate the total calculation, in which we terminate the reordering procedure appropriately instead of reordering to the end. Numerical experiments show the effectiveness of this adaptive reordered method.

1. Introduction

As the Internet develops rapidly, the development of web search engines is becoming more and more important. As one of the most popular search engines, Google attributes the famous PageRank algorithm to compute the PageRank vector, which amounts to computing the stationary distribution of the transition possibility matrix that describes the web graph with the power method. In the web graph, there are many dangling nodes, that is, pages without links to the other pages. These dangling nodes could trigger storage and computational problems during the PageRank computation [1].

First, let us introduce some principles and symbols in the PageRank problem. The web structure with n nodes can be expressed in the form of a spares hyperlink matrix $\mathbf{H}$. The (i, j)th element of $\mathbf{H}$ is $1/O_i$ if there is a link from node i to node j, and 0 otherwise, where O_i is the number of the node i's outlink. This spares matrix is called transition possibility matrix. The row sums are 1 for the nondangling nodes and 0 for the dangling nodes. The PageRank vector is the stationary distribution for the Markov Chain related to $\mathbf{H}$. With the existence of the dangling nodes, there are rows with all 0 entries, making $\mathbf{H}$ not stochastic. To remedy this, one can replace the $\mathbf{0}^T$ rows with $\mathbf{v}^T = \mathbf{e}^T/n$, called personalization vector, where $\mathbf{e}$ is the vector of all ones. This replacement gives

$$\mathbf{Q} = \mathbf{H} + \mathbf{a}\mathbf{v}^T, \tag{1}$$

where $\mathbf{a}$ is an indexing vector with its element $a_i = 1$ if row i of $\mathbf{H}$ is $\mathbf{0}^T$ row. To guarantee the existence and uniqueness of the PageRank vector, one more rank-1 update is needed, which yields the stochastic and irreducible Google matrix

$$\mathbf{G} = \alpha\mathbf{Q} + (1 - \alpha)\,\mathbf{e}\mathbf{v}^T, \tag{2}$$

where $0 < \alpha < 1$. Then, the power method is used on $\mathbf{G}$ to calculate the stationary vector, that is, the PageRank vector [2]. Due to the slow convergence rate of the power method, many acceleration methods have been proposed, such as the extrapolation methods [3], the adaptive method [4], and some other numerical methods [5–11].

With the large amount of the dangling nodes in the web, there are many $\mathbf{0}^T$ rows in the hyperlink matrix $\mathbf{H}$. It is worthwhile to consider methods to take advantage of these identical rows. Lee et al. give a two-stage algorithm to improve the PageRank computation by lumping the dangling nodes and aggregating the nondangling nodes [12]. By recursively

using the lumping procedure, Langville and Meyer propose a reordered algorithm [13]. In their algorithm, the top left submatrix gets smaller as the reordering procedure proceeds, by recursively reordering the $\mathbf{0}^T$ rows of the top left submatrix to the bottom, until there is no $\mathbf{0}^T$ rows in the new top left submatrix. The stationary vector of the final top left submatrix is easy to compute since its dimension is much smaller compared to the origin matrix. Then, forward substitutions are carried on to get the full PageRank vector.

Although it much easier to calculate the PageRank vector with a reduced matrix, the continual reorderings are consuming. In this paper, we try to find a compromise between getting a reduced submatrix and saving the calculation spent on reordering. We give a threshold for the recursively reordered method and terminate the reordering procedure halfway, which we call the adaptive reordered method.

This paper is organized as follows. We briefly review the reordered method [13] in Section 2. In Section 3, an improved reordered method with an adaptive stopping criterion is presented. Numerical experiments on two realistic matrices are presented using both original reordered method and adaptive reordered method in Section 4. Conclusions and future work are presented in Section 5.

2. Reordered Method for the PageRank Problem

In this section, we briefly introduce the reordered method given by Langville and Meyer [13].

Theorem 2.1 in [13] reveals the relationship between the PageRank vector and the solution to the previous linear system that solving

$$\mathbf{x}^T(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^T \tag{3}$$

and letting $\boldsymbol{\pi}^T = \mathbf{x}^T/\mathbf{x}^T\mathbf{e}$ produces the PageRank vector.

On the other side, the nodes of web can be classified into dangling nodes (D) and nondangling nodes (ND). In the hyperlink matrix $\mathbf{H}$, the rows corresponding to dangling nodes are all $\mathbf{0}^T$ rows. By permuting the rows and columns of $\mathbf{H}$, we can achieve the aim that all the $\mathbf{0}^T$ rows are at the bottom of $\mathbf{H}$. As each of dangling nodes and nondangling nodes corresponds to its particular row, one can permute the rows of $\mathbf{H}$ and then get $\mathbf{P}$, whose rows corresponding to dangling nodes are all at the bottom:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{4}$$

where $\mathbf{P}_{11}$ represents the links among nondangling nodes and $\mathbf{P}_{12}$ represents the links from nondangling nodes to dangling nodes. The zero rows at the bottom correspond to the dangling nodes. According to the definition of hyperlink matrix, we can draw the conclusions that $\mathbf{P}_{11} \geq 0$, $\mathbf{P}_{12} \geq 0$, and the row sums of $\mathbf{P}_{11}$ and $\mathbf{P}_{12}$ are all equal to 1. Then, the coefficient matrix in the previous linear system is

$$(\mathbf{I} - \alpha\mathbf{P}) = \begin{pmatrix} \mathbf{I} - \alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{5}$$

and its inverse is

$$(\mathbf{I} - \alpha\mathbf{P})^{-1} = \begin{pmatrix} (\mathbf{I} - \alpha\mathbf{P}_{11})^{-1} & \alpha\left((\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\right)\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{6}$$

Then, the solution $\mathbf{x}^T$ to the linear system $\mathbf{x}^T(\mathbf{I}-\alpha\mathbf{P}) = \mathbf{v}^T$ can be written as

$$\begin{aligned} \mathbf{x}^T &= \mathbf{v}^T(\mathbf{I} - \alpha\mathbf{P})^{-1} \\ &= \left(\mathbf{v}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1} \mid \alpha\mathbf{v}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12} + \mathbf{v}_2^T\right), \end{aligned} \tag{7}$$

where the personalization vector $\mathbf{v}^T$ has been partitioned into nondangling and dangling parts, corresponding to $\mathbf{v}_1^T$ and $\mathbf{v}_2^T$ separately.

Based on the previous idea, Langville and Meyer come out with a method that recursively permutes the top left block until the final $\mathbf{P}_{11}$ has no zero rows. The recursive permutations make the dimension of $\mathbf{P}_{11}$ smaller than just permuting once. Eventually, one can get a matrix with the following form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1b} \\ & \mathbf{0} & \mathbf{P}_{23} & \cdots & \mathbf{P}_{2b} \\ & & \mathbf{0} & \cdots & \mathbf{P}_{3b} \\ & & & \ddots & \vdots \\ & & & & \mathbf{0} \end{pmatrix}, \tag{8}$$

where $b \geq 2$ is the number of blocks of each row. Then, the corresponding coefficient matrix of linear system is

$$(\mathbf{I} - \alpha\mathbf{P}) = \begin{pmatrix} \mathbf{I} - \alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} & -\alpha\mathbf{P}_{13} & \cdots & -\alpha\mathbf{P}_{1b} \\ & \mathbf{I} & -\alpha\mathbf{P}_{23} & \cdots & -\alpha\mathbf{P}_{2b} \\ & & \mathbf{I} & \cdots & -\alpha\mathbf{P}_{3b} \\ & & & \ddots & \vdots \\ & & & & \mathbf{I} \end{pmatrix}. \tag{9}$$

This reordered method can be described in Algorithm 1.

This reordered method reduced the computation of the PageRank vector that one just needs to solve a much smaller linear system, $\mathbf{x}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$, and the forward substitution in the third step gives the full PageRank vector. However, the recursively reordering procedure in the first step is time consuming and may bring too much overhead if b, the number of blocks, is too large. So it is imperative to introduce some mechanism to overcome this drawback, which we will describe in the following section.

3. Adaptive Reordered Method for the PageRank

3.1. Adaptive Reordered Method. In this section, we introduce an adaptive reordered method. This method is based on

(1) Reorder the hyperlink matrix so that in the top left submatrix all the zero rows are at the bottom in every reordering procedure, until there is no $\mathbf{0}^T$ rows in the new top left submatrix.
(2) Solve $\mathbf{x}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$ using Jacobi method.
(3) For $i = 2$ to b, Compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{P}_{ji} + \mathbf{v}_i^T$.
(4) Compute $\boldsymbol{\pi}^T = \left[\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T\right] / \left\|\left[\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T\right]\right\|_1$.

Algorithm 1: The reordered algorithm [13].

(1) Reorder the hyperlink matrix so that in the top left submatrix all the zero rows are at the bottom in every reordering procedure, until the given stopping criterion is reached.
(2) Solve $\mathbf{x}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$ using Jacobi method.
(3) For $i = 2$ to b, Compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{P}_{ji} + \mathbf{v}_i^T$.
(4) Compute $\boldsymbol{\pi}^T = \left[\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T\right] / \left\|\left[\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T\right]\right\|_1$.

Algorithm 2: The adaptive reordered algorithm.

Langville and Meyer's work in [13] and utilizes a stopping criterion that once the criterion is reached, the reordering procedure will be terminated. The adaptive reordered method can both get rid of the potential overhead brought by the recursive reordering and keep the merit of reduction of computation. The adaptive reordered algorithm is shown in Algorithm 2.

According to Langville and Meyer's work, the merit of the reordered method lies in the second step of Algorithm 1 that the computation is reduced because of the obtained smaller linear system. But the consuming recursively reordering procedure could bring too much overheads that offset this merit, or even worse. So, we consider to bring in a stopping criterion to make a compromise between reordering's merit and overhead.

3.2. Analysis of the Adaptive Reordered Method. From the merit-and-overhead analysis previous, we can get the idea that once the overhead of the first step exceeds the computational reduction brought by the smaller linear system in the second step, the recursively reordering procedure should be terminated. We assume that, before deciding whether the next reordering procedure is necessary, the rank of the top left block is r_1, and after the following reordering step, the rank of the new top left block is r_2. Apparently, $r_1 \geq r_2$. The new reordering step will bring an extra reordering expense which is approximately $O(r_1^2)$, an extra reordering expense which is $O(r_2(r_1 - r_2))$ approximately. It also brings the computational reduction which is approximately $O(130(r_1^2 - r_2^2))$, in which 130 is the approximate times of matrix-vector products in Jacobi method before the result reaches the precision demand. With these analyses, we can obtain the following stopping criterion.

Stopping Criterion. Once $130(r_1^2 - r_2^2) \leq r_1^2 + r_2(r_1 - r_2)$, the reordering procedure in the first step of Algorithm 2 stops.

With this stopping criterion, our adaptive reordered method would stop at the appropriate time, which both reduces the dimension of the linear system to be solved and saves the excessive expense of the recursively reordering procedure.

4. Numerical Experiments

In this section, we present the numerical experiments to compare the adaptive reordered method to the original reordered method [13]. The experiments are on two matrices. The first matrix is *wb-cs-stanford.mat* [14], which contains 9914 pages, and the second matrix is *Stanford.mat* [15], which contains 281903 pages. Both matrices are direct graph and before the experiments they are firstly turned into row-stochastic matrices that all the row sums are 1. Similar to [13], we also choose Jacobi method to solve the linear system. We use $\alpha = 0.85$ as the damping factor and $\tau = 10^{-10}$ as the convergence tolerance. The numerical experiments are carried out on MATLAB. We call the reordered method and the adaptive reordered method *RD* and *aRD* for short. The performances of both methods are as follows.

Figures 1 and 2 show the structures of the original matrix and those being reordered using *RD* and *aRD*. One can observe that the rough shapes of the matrices reordered by both methods are similar, which means in both methods that the dimensions of the linear systems to be solved in the second step are almost equal and the expenses of solving the linear systems are close. Actually, in the experiment on *wb-cs-stanford.mat*, the dimension of $\mathbf{P}_{11}$ when using *aRD* is 6592 and that is 6585 when using *RD*.

Table 1 shows b and the time spent on the reordering step of the two methods applied on the two matrices. In this table, $t1$ represents the time spent on reordering step and $t2$ the time on the remaining steps, and t is the sum of $t1$

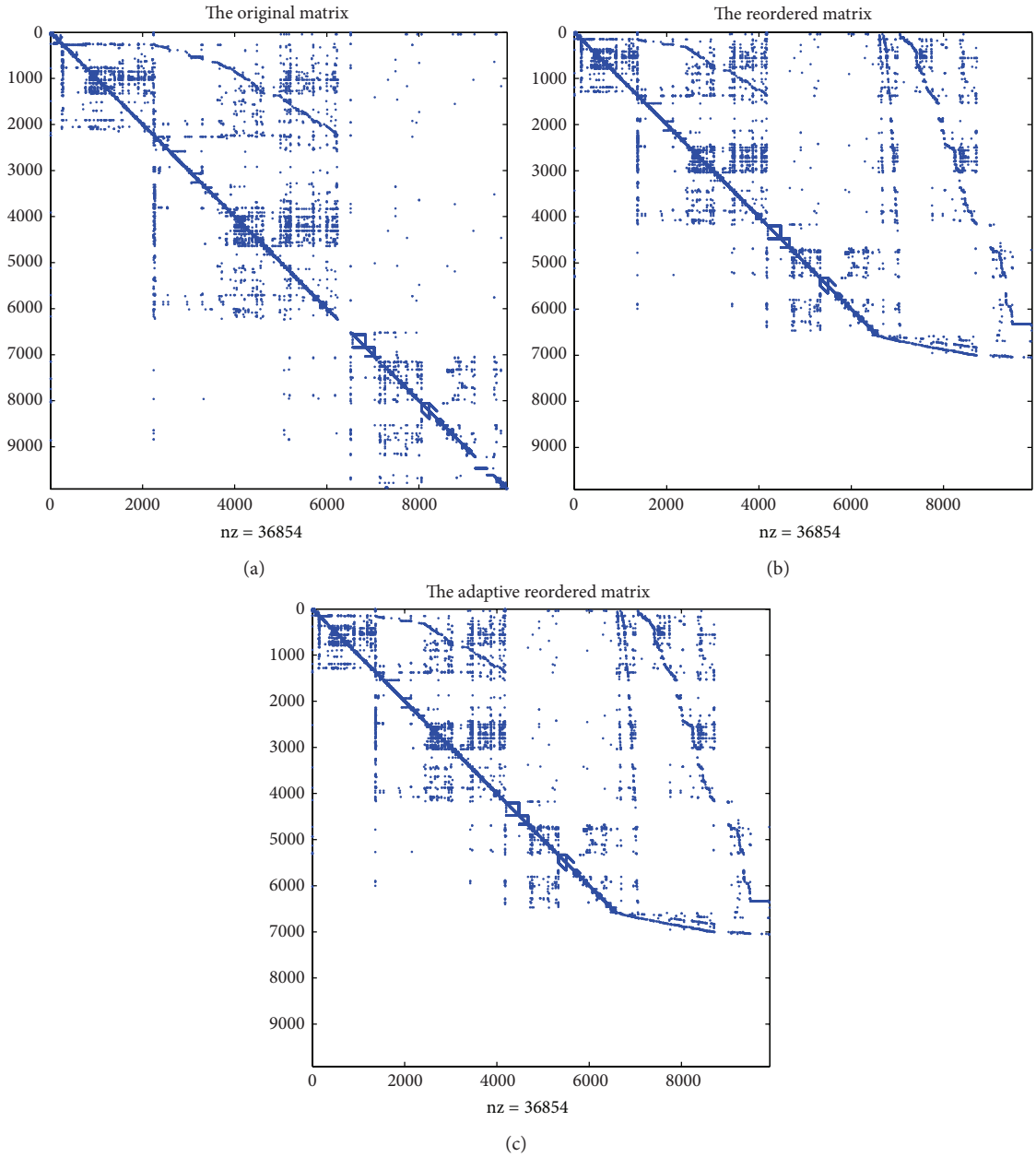

FIGURE 1: The resulted image of the original *wb-cs-stanford.mat* (a) and matrices reordered using *RD* and *aRD* ((b) and (c)).

and $t2$. With respect to *wb-cs-stanford.mat*, there are four blocks if *aRD* is used and their sizes are 6592, 88, 356, and 2861 successively, while there are seven blocks when *RD* is used and their sizes are 6585, 3, 4, 17, 88, 356, and 2861. Comparing these two sets of data, one can tell that, when using *RD*, the follow-up reorderings that get the subdivided blocks, 6585, 3, 4, and 17, does not reduce the dimension of the top left block substantially, which means that these reorderings are not very useful. Our adaptive reordered method also saves much expense when it is applied to *Stanford.mat*. There are 141 blocks when using *RD* while there are only 3 blocks when using *aRD*. After reviewing the data, we find that the sizes of blocks of *aRD* are 258177, 3338, and 20315 and those of *RD* are 257824, ..., 14, 73, 3338, and 20315, where the sizes omitted are all trivial and the reorderings spent on those blocks are not worthy. As one more block brings in one more reordering operation, the data of this table shows that *aRD* saves much more expense of reordering procedure than *RD* does.

5. Conclusion

In this paper, we introduce an adaptive reordered method for solving the PageRank problem. Comparing to the original reordered method [13], we bring in an effective stopping criterion which can both get rid of the overhead brought by the recursively reordering procedure and keep its fast computational speed. Numerical results with two examples

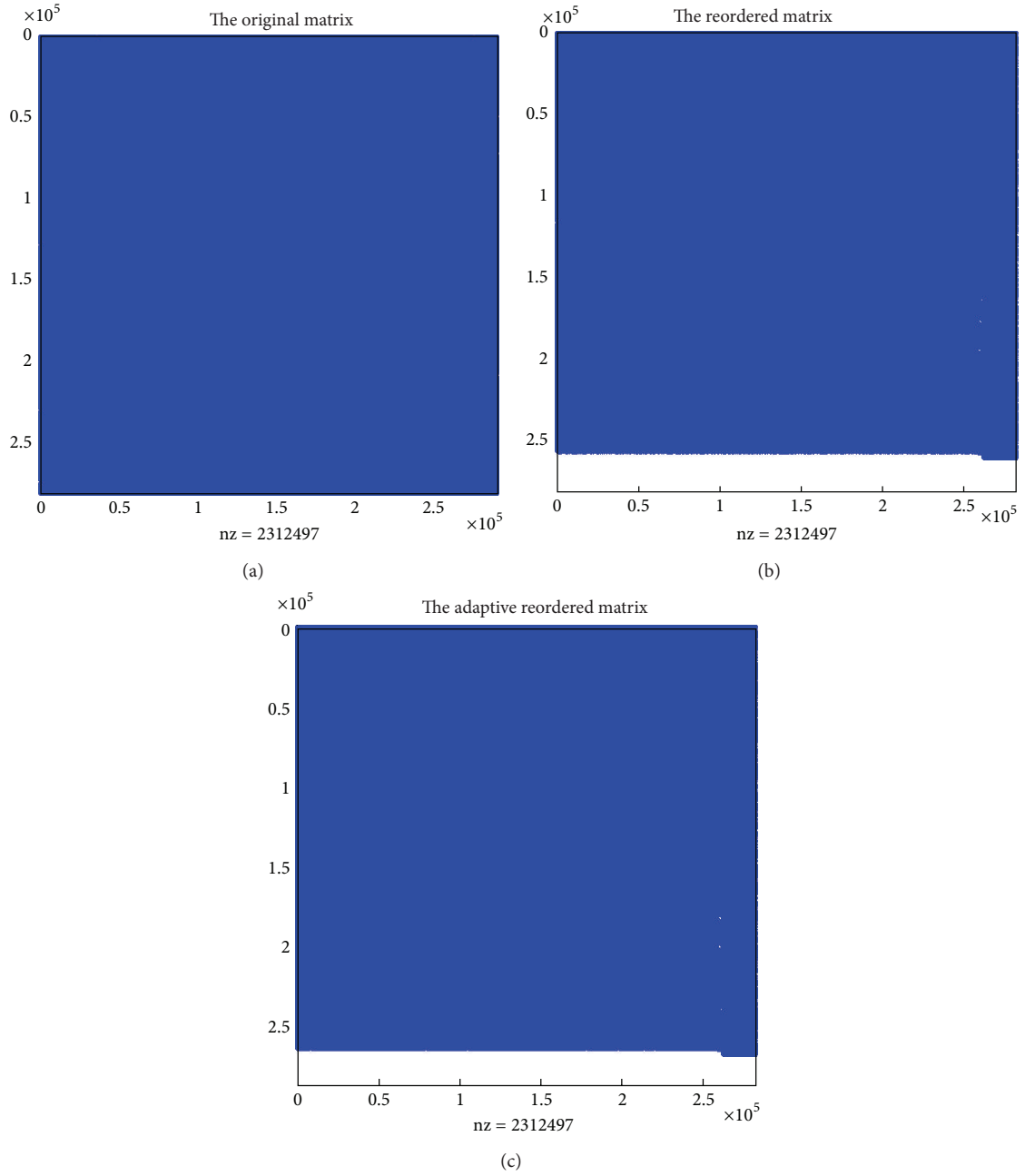

Figure 2: The resulted image of the original *Stanford.mat* (a) and matrices reordered using *RD* and *aRD* ((b) and (c)).

Table 1: Comparison of *RD* and *aRD*.

		wb-cs-stanford	*Stanford*
RD	b	7	141
	$t1$	0.0315	66.95
	$t2$	0.1250	17.09
	t	0.1565	84.04
aRD	b	4	3
	$t1$	0.0184	1.45
	$t2$	0.1100	9.13
	t	0.1284	10.58

demonstrate the good performance of this method. Jacobi method is utilized in this paper, and meanwhile other acceleration methods and preconditioning approaches can also be utilized together, which could enhance the effect of the reordered method. These are topics that deserve to be studied and will be our work in the future.

Acknowledgments

This research is supported by 973 Program (2013CB329404) and Chinese Universities Specialized Research Fund for the Doctoral Program (20110185110020).

References

[1] S. Brin, L. Page, R. Motwami, and T. Winograd, "The PageRank citation ranking:bringing order to the web," Tech. Rep. 1999-0120, Computer Science Department, Stanford University, Stanford, Calif, USA, 1999.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[3] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolationmethods for accelerating PageRank computations," in *Proceedings of the 12th International World Wide Web Conference*, ACM Press, New York, NY, USA, 2003.

[4] S. D. Kamvar, T. H. Haveliwala, and G. H. Golub, "Adaptive methods for the computationof PageRank," Tech. Rep. 2003-26, Stanford University, Stanford, Calif, USA, 2003.

[5] Y. Lin, X. Shi, and Y. Wei, "On computing PageRank via lumping the Google matrix," *Journal of Computational and Applied Mathematics*, vol. 224, no. 2, pp. 702–708, 2009.

[6] G. Wu and Y. Wei, "A Power-Arnoldi algorithm for computing PageRank," *Numerical Linear Algebra with Applications*, vol. 14, no. 7, pp. 521–546, 2007.

[7] G. Wu and Y. Wei, "Arnoldi versus GMRES for computing PageRank: a theoreticalcontribution to Google's PageRank problem," *ACM Transactions on Information Systems*, vol. 28, no. 3, article 11, 2010.

[8] G. Wu and Y. Wei, "An Arnoldi-extrapolation algorithm for computing PageRank," *Journal of Computational and Applied Mathematics*, vol. 234, no. 11, pp. 3196–3212, 2010.

[9] G. Wu, Y. Zhang, and Y. Wei, "Krylov subspace algorithms for computing GeneRank for the analysis of microarray data mining," *Journal of Computational Biology*, vol. 17, no. 4, pp. 631–646, 2010.

[10] Q. Yu, Z. Miao, G. Wu, and Y. Wei, "Lumping algorithms for computing Google's PageRank and its derivative, with attention to unreferenced nodes," *Information Retrieval*, vol. 15, no. 6, pp. 503–526, 2012.

[11] G. Wu, W. Xu, Y. Zhang, and Y. Wei, "A preconditioned conjugate gradient algorithm for GeneRank with application to microarray data mining," *Data Mining and Knowledge Discovery*, vol. 26, no. 1, pp. 27–56, 2013.

[12] C. P. C. Lee, G. H. Golub, and S. A. Zenios, "A fast two-stage algorithm for computingPageRank and its extensions," Tech. Rep. SCCM-2003-15, Scientific Computationand Computational Mathematics, Stanford University, Stanford, CA, USA, 2003.

[13] A. N. Langville and C. D. Meyer, "A reordering for the PageRank problem," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 2112–2120, 2006.

[14] http://www.cise.ufl.edu/research/sparse/matrices/Gleich/wb-cs-stanford.html.

[15] http://www.cise.ufl.edu/research/sparse/matrices/Kamvar/Stanford.html.