Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 438987, 8 pages
http://dx.doi.org/10.1155/2013/438987

Research Article

An Improved Approach to the PageRank Problems

Yue Xie,[1,2] Ting-Zhu Huang,[1] Chun Wen,[1] and De-An Wu[1]

[1] *School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*
[2] *Department of Mathematics, The University of Hong Kong, Pokfulam, Hong Kong*

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

Received 1 August 2013; Revised 25 November 2013; Accepted 25 November 2013

Academic Editor: Zhongxiao Jia

We introduce a partition of the web pages particularly suited to the PageRank problems in which the web link graph has a nested block structure. Based on the partition of the web pages, dangling nodes, common nodes, and general nodes, the hyperlink matrix can be reordered to be a more simple block structure. Then based on the parallel computation method, we propose an algorithm for the PageRank problems. In this algorithm, the dimension of the linear system becomes smaller, and the vector for general nodes in each block can be calculated separately in every iteration. Numerical experiments show that this approach speeds up the computation of PageRank.

1. Introduction

The rapid growth of the World Wide Web has created a need for search tools. One of the best-known algorithms in web search is Google's PageRank algorithm [1]. Google's PageRank algorithm is based on a random surfer model [1] which can be viewed as a stationary distribution of a Markov chain. Simultaneously with the random surfer model, a different but closely related approach, the HITS algorithm, was invented in [2]. Another model SALSA [3] incorporated ideas from both HITS and PageRank to create another ranking of webpages.

In this paper, we focus on Google's PageRank algorithm. Let us introduce some notations about Google's PageRank algorithm. We can model the web as a directed graph with the web pages as the nodes and the hyperlinks as the directed edges. In the graph, if there is a link from page P_i to page P_j, then, for page P_i, it has an outlink to page P_j, and, for page P_j, it has an inlink from page P_i. Then we can define the elements of a hyperlink matrix H as follows.

If the web page P_i has outlinks $Q_i \geq 1$, then, for each link from page P_i to another page P_j, the element $h_{i,j}$ of the matrix H is $1/Q_i$. If there is no link from page P_i to page P_j, then the element $h_{i,j}$ of H is 0. The scalar Q_i is the number of outlinks from the page P_i. Thus, each nonzero row of H sums to 1. If the page P_i has no outlinks at all (such as a pdf, image, or audio file), it is called a dangling node, and all elements in the ith row of H are set to 0.

The problem is that if at least one node has zero outdegree, that is, no outlinks, then the Markov chain is absorbing, so a modification to H is needed. In order to resolve this, the founders of Google, Brin and Page suggest replacing each zero row (corresponding to a dangling node) of the sparse hyperlink matrix with a dense nonnegative vector v^T ($v^T e = 1$; e is the column vector of all ones and v^T also could be a personalized vector, see [4, 5]) and create the new stochastic matrix denoted by S, $S = H + dv^T$. In the vector d, the element $d_i = 1$ if the ith row of H corresponds to a dangling node, and 0 otherwise. Another problem is that there is nothing in our definition so far that guarantees the convergence of the PageRank algorithm or the uniqueness of the PageRank vector with the matrix S. In general, if the matrix S is irreducible, this problem can be settled. Thus, Brin and Page added another dense perturbation matrix ev^T that creates direct connections between each page to force the matrix to be irreducible. Then, the stochastic, irreducible matrix is called the Google matrix G and given by

$$G = \alpha S + (1-\alpha) ev^T = \alpha H + \alpha d v^T + (1-\alpha) ev^T, \quad (1)$$

where $0 < \alpha < 1$ (a typical value for α is between 0.85 and 0.95. It is shown in [6] that α controls the convergence rate of the PageRank algorithm). Mathematically, the PageRank vector π is the stationary distribution of the so-called Google matrix G.

Now, we have got many methods for solving the PageRank vector π, such as the famous Power Method [1, 7, 8]. Due to the sheer size of the web (over 3 billion pages), this computation can take several days. In [9], Arasu et al. used values from the current iteration as they become available, rather than using only values from the previous iteration. They also suggested that exploiting the "bow-tie" structure of the web [10] would be useful in computing PageRank. In [11], Kamvar et al. presented a variety of extrapolation methods. In [12], Avrachenkov et al. showed that Monte Carlo methods already provide good estimation of the PageRank for relatively important pages after one iteration. Gleich et al. in [13] presented an inner-outer iterative algorithm for accelerating PageRank computations. To put it another way, for the existence of the dangling nodes, Lee et al. [14] partitioned the web into dangling and nondangling nodes and applied an aggregation method to this partition.

Recently, the structure of the web link graph has been noticed. Kamvar et al. in [4] brilliantly exploited the block structure of the web for computing PageRank. They also exploited the fact that pages with lower page rank tend to converge faster and propose adaptive methods in [15]. Based on the characteristics of the web link graph, research on parallelization of PageRank can be found in [16–21]. In [21], Manaskasemsak and Rungsawang discussed a parallelization of the power method. In [17], Gleich et al. introduced a method to compare the various linear system formulations in terms of parallel runtime performance. Cevahir et al. in [16] proposed the site-based partitioning and repartitioning techniques for parallel PageRank computation. Some special models for parallel PageRank were proposed in [18–20].

In our paper, we combine ideas from the existence of the dangling nodes and the block structure of the web and exploit a new structure for the hyperlink matrix H. Then some parallel computation methods are applied to speed up the computation of PageRank by using a partition of the nodes. Firstly, we present that our target is to compute the PageRank of the nondangling nodes in the linear system for the Google problem [22] (Section 2). Secondly, according to the partition of the web pages, we get a special structure of the hyperlink matrix, and then we propose an algorithm (Section 3). Finally, we make an analysis of our algorithms, and some numerical results are given (Sections 4 and 5).

2. The Problem

Generally, the Google problem is to solve the eigenvector π of the matrix G in the following equation:

$$\pi^T = \pi^T G, \quad \pi \geq 0, \quad \left\|\pi^T\right\|_1 = 1. \tag{2}$$

Here, we introduce some theorems to show that the Google problem can turn out to be a linear system problem and only need to compute the unnormalized PageRank subvector of the nondangling nodes. In the following, the matrix I denotes the identity matrix.

Theorem 1 (see [22, linear system for Google problem]). *Suppose that the matrix H is a hyperlink matrix. Solving the linear system,*

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

and letting $\pi^T = x^T / \|x\|_1$ produce the PageRank vector.

Since the coefficient matrix $(I - \alpha H)$ in (3) is an M-matrix (Theorem 8.(4.2) in [23]) as well as nonsingular and irreducible, thus, the solution of the linear system in Theorem 1 is existent and unique.

The rows in the matrix H corresponding to the dangling nodes would be zero. It is natural as well as efficient to exclude the dangling nodes from the PageRank computation. This can be done by partitioning the web nodes into nondangling nodes and dangling nodes. This is similar to the method of "lumping" all the dangling nodes into a single node [24]. Supposing that the rows and columns of H are permuted corresponding to the partition, then the rows corresponding to the dangling nodes are at the bottom of the matrix:

$$H = \begin{array}{c c} & \begin{array}{cc} ND & D \end{array} \\ \begin{array}{c} ND \\ D \end{array} & \begin{bmatrix} \widehat{H}_{11} & \widehat{H}_{12} \\ 0 & 0 \end{bmatrix} \end{array}, \tag{4}$$

where ND is the set of the nondangling nodes and D is the set of the dangling nodes.

Then, the coefficient matrix $(I - \alpha H)$ in (3) becomes

$$(I - \alpha H) = \begin{bmatrix} I - \alpha \widehat{H}_{11} & -\alpha \widehat{H}_{12} \\ 0 & I \end{bmatrix}, \tag{5}$$

and the inverse of this matrix is

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

Therefore, the unnormalized PageRank vector $x^T = v^T (I - \alpha H)^{-1}$ in (4) can be written as

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

Then, Langville and Meyer [22] proposed two reordered PageRank algorithms for computing the PageRank vector. One is Algorithm 1, called reordered PageRank algorithm, and the other is called reordered PageRank algorithm. However, unfortunately, the reordered PageRank algorithm is not necessarily an improvement over Algorithm 1 in some cases.

In this reordered PageRank Algorithm 1, the only system that must be solved is $\hat{x}_1^T (I - \alpha \widehat{H}_{11}) = \hat{v}_1^T$. The reordered PageRank Algorithm 2 is based on a process of locating zero rows which can be repeated recursively on smaller and smaller submatrices of $\widehat{H}_{11}$, continuing until a submatrix is created that has no zero rows. For interested readers, the detail of the reordered PageRank algorithms can be found in [22]. However, this structure of the web they

```
(1) Partition the web nodes into dangling and nondangling nodes, so that the hyperlink
    matrix $H$ has the structure of (4).
(2) Solve $\widehat{x}_1^T$ from $\widehat{x}_1^T\left(I - \alpha\widehat{H}_{11}\right) = \widehat{v}_1^T$.
(3) Compute $\widehat{x}_2^T = \alpha\widehat{x}_1^T\widehat{H}_{12} + \widehat{v}_1^T$.
(4) $\pi^T = \left(\widehat{x}_1^T, \widehat{x}_2^T\right)/\left\|\widehat{x}_1^T, \widehat{x}_2^T\right\|_1$.
```

Algorithm 1: Reordered PageRank Algorithm [22].

```
(1) Partition the web nodes which form $m$ blocks: $S = (S_1, S_2, \dots, S_m)$ into $m + 2$
    blocks: $S = (S_1^2, S_2^2, \dots, S_m^2, CN, D)$, so the hyperlink matrix $H$ has the structure of (12).
(2) Partition the given vector $v^T = \left(\widehat{v}_1^T, \widehat{v}_2^T\right)$ and PageRank vector $x^T = \left(\widehat{x}_1^T, \widehat{x}_2^T\right)$
    according to the size of the $m + 2$ blocks:
        $\widehat{v}_1^T = \left(w_1^T, w_2^T\right), w_1^T = \left(v_1^T, v_2^T, \dots, v_m^T\right),\ w_2^T = \left(v_{m+1}^T\right), \widehat{v}_2^T = \left(v_{m+2}^T\right);$
        $\widehat{x}_1^T = \left(y_1^T, y_2^T\right), y_1^T = \left(x_1^T, x_2^T, \dots, x_m^T\right), y_2^T = \left(x_{m+1}^T\right), \widehat{x}_2^T = \left(x_{m+2}^T\right).$
(3) Compute the limiting vector of $y_1^T$ by iterations as follow:
    (a) Compute $\left(r_1^T, r_2^T, \dots, r_m^T\right) = y_1^{(k-1)T}B(I - \alpha F)^{-1}E + w_2^T(I - \alpha F)^{-1}E + w_1^T$;
    (b) Solve for $y_1^{(k)T} = \left(x_1^{(k)T}, x_2^{(k)T}, \dots, x_m^{(k)T}\right)$ in
            $x_i^{(k)T}\left(I - \alpha H_{ii}\right) = r_i^T, \quad i = 1, 2, \dots, m.$
(4) Compute
        $y_2^T = \left(w_2^T + y_1^T\alpha B\right)(I - \alpha F)^{-1},$
        $\widehat{x}_2^T = \alpha\widehat{x}_1^T\widehat{H}_{12} + \widehat{v}_2^T = \alpha\left(y_1^T, y_2^T\right)\widehat{H}_{12} + \widehat{v}_2^T.$
(5) Normalize $\pi^T = \left(y_1^T, y_2^T, \widehat{x}_2^T\right)/\left\|\left(y_1^T, y_2^T, \widehat{x}_2^T\right)\right\|_1$.
```

Algorithm 2: An algorithm based on a separation of the common nodes.

exploit in reordered PageRank Algorithm 2 is not practical, as reordering the web matrix according to this structure requires depth-first search, which is prohibitively costly on the web. To put it another way, even though some hyperlink matrices H can be suited to the reordered PageRank algorithm, the structure may not exist for some hyperlink matrices. Thus the reordered PageRank Algorithm 2 will have no advantage over Algorithm 1 in this worst case. Similarly, we can find the same conclusion in their experiments. Thus, we come back to (4) and reorder the structure of the matrix $\widehat{H}_{11}$ to speed up the computation of PageRank vector. The objective function becomes

$$\widehat{x}_1^T\left(I - \alpha\widehat{H}_{11}\right) = \widehat{v}_1^T, \tag{8}$$

where the coefficient matrix $(I - \alpha\widehat{H}_{11})$ is the nontrivial leading principal submatrix of $(I - \alpha H)$ and it is nonsingular (Theorem 6.(4.16) of [23]).

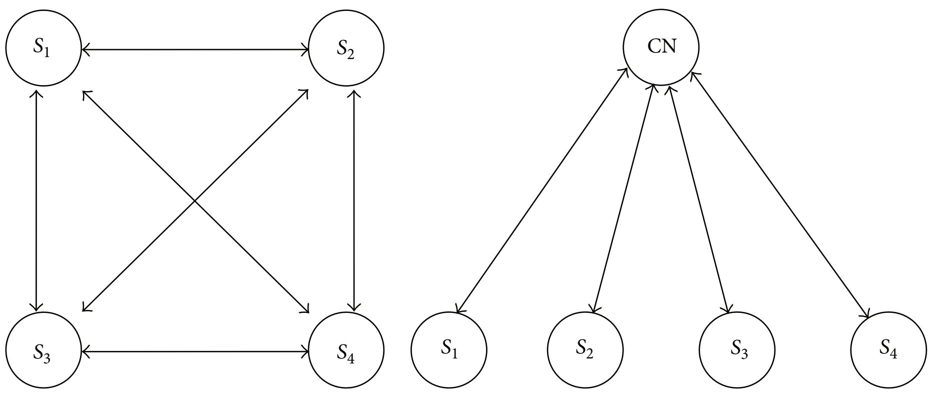
3. PageRank Algorithms Based on a Separation of the Common Nodes

3.1. The Block Structure of the Web. It is noted in [4] that when sorted by Uniform Resource Location (URL), the web link graph has a nested block structure: the vast majority of hyperlinks link pages on a host to other pages on the same host. This property was demonstrated by examination on realistic datasets. So in the following sections, we consider the webs that have block structure. To simplify notation, without loss of generality, we will assume that a web link graph has a block structure of m blocks: $S_1, S_2, \dots, S_m$. So the hyperlink matrix H is

$$H = \begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_m \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & \cdots & S_m \end{array} \\ \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1,m} \\ H_{21} & H_{22} & \cdots & H_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ H_{m,1} & H_{m,2} & \cdots & H_{m,m} \end{bmatrix} \end{array}. \tag{9}$$

Then, we separate the dangling nodes from each of the blocks. Thus, we get the new blocks S_k^1, $k \in (1, \dots, m)$, which are the original blocks S_k with dangling nodes removed. The set of nodes $S = (S_1, S_2, \dots, S_m)$ is $S = (ND, D)$, where $ND = (S_1^1, S_2^1, \dots, S_m^1)$ and D is the set of the dangling nodes. The rows and columns of H can be permuted, making the rows corresponding to the dangling nodes at the bottom of the matrix just like (4) in Section 2:

$$\begin{aligned} H &= \begin{array}{c} \\ ND \\ D \end{array} \begin{array}{c} \begin{array}{cc} ND & D \end{array} \\ \begin{bmatrix} \widehat{H}_{11} & \widehat{H}_{12} \\ 0 & 0 \end{bmatrix} \end{array} \\ &= \begin{array}{c} \\ S_1^1 \\ S_2^1 \\ \vdots \\ S_m^1 \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} S_1^1 & S_2^1 & \cdots & S_m^1 & D \end{array} \\ \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1,m} & H_{1,m+1} \\ H_{21} & H_{22} & \cdots & H_{2,m} & H_{2,m+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ H_{m,1} & H_{m,2} & \cdots & H_{m,m} & H_{m,m+1} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \end{aligned} \tag{10}$$



(a) The structure before the separation

(b) The structure after the separation

Figure 1: A separation of the common nodes for a web link graph which has four blocks.

In the above equation, the submatrix $\widehat{H}_{11}$ is

$$\widehat{H}_{11} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1,m} \\ H_{21} & H_{22} & \cdots & H_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ H_{m,1} & H_{m,2} & \cdots & H_{m,m} \end{bmatrix}. \qquad (11)$$

3.2. A Separation of the Common Nodes. To investigate the detail of the web structure, we can see the experiments in [4]. They used LARGEWEB link graph [25] and considered the version of LARGEWEB with dangling nodes removed, which contains roughly 70 M nodes, with over 600 M edges, and requires 3.6 GB of storage. They partitioned the links in the graph into "intrahost" links, which means links from a page to another page in the same host, and "interhost" links, which means links from a page to a page in a different host. Through counting the number of the two different links separately, Table 2 in [4] shows that 93.6% of the links in the datasets are intrahost links and 6.4% are interhost links, which means that larger majority of links are intrahost links and only a minority of links are interhost links. They also found the same result by partitioning the links according to different domains. This result leads to a deeper study of the structure of the hyperlink matrix H. That is, if the pages are grouped by domain, host, or others, the graph for the pages will appear as a block structure. Then in each subblock, a minority of nodes have links to other blocks, and in this paper we call them common nodes. The definition of common node is given as follows.

Definition 2 (common node). Assume that a web link graph with dangling nodes removed has n blocks $S_1, S_2, \ldots, S_n$. If a node in a block S_i $(1 \leq i \leq n)$ has at least one outlink to another different block S_j $(j \neq i,\ 1 \leq j \leq n)$ or inlink from another different block S_j $(j \neq i, 1 \leq j \leq n)$, we call it common node.

If a node in a web link graph is not a dangling node or a common node, then we call it general node. The nodes in a web link graph are divided into three classes: dangling node, common node, and general node. Specially, the common nodes and general nodes belong to the nondangling nodes.

There is no dangling node in the blocks $S_1^1, S_2^1, \ldots, S_m^1$, so we consider separating all the common nodes from the blocks $S_1^1, S_2^1, \ldots, S_m^1$ and form a new block denoted by CN. Hence, the set of nodes $S = (S_1^1, S_2^1, \ldots, S_m^1, D)$ is $S = (S_1^2, S_2^2, \ldots, S_m^2, CN, D)$. The new block S_k^2 $(1 \leq k \leq m)$ is the block S_k^1 with common nodes removed. Thus, any hyperlink submatrix $H_{i,j}$ $(i \neq j)$ corresponding to two different blocks S_i^2 and S_j^2 becomes zero matrix because there are no interlinks between different blocks in $S_1^2, S_2^2, \ldots, S_m^2$.

In Figure 1, a simple example is shown to illustrate the change after a separation of the common nodes. In Figure 1(a), there are four blocks S_1, S_2, S_3, and S_4 in a web link graph, and each of them has links to others. However, in Figure 1(b), after separating the common nodes from the four blocks and lumping the common nodes into a block denoted by CN, there are no links among the four new blocks. The links exist only between the CN and the four new blocks. Once the above is done, the hyperlink matrix H corresponding to the partition of the web nodes, $S = (S_1^2, S_2^2, \ldots, S_m^2, CN, D)$, has the following structure:

$$H = \begin{array}{c} \\ S_1^2 \\ S_2^2 \\ \vdots \\ S_m^2 \\ CN \\ D \end{array} \begin{array}{c} \begin{array}{cccccc} S_1^2 & S_2^2 & \cdots & S_m^2 & CN & D \end{array} \\ \begin{bmatrix} H_{11} & 0 & \cdots & 0 & H_{1,m+1} & H_{1,m+2} \\ 0 & H_{22} & \cdots & 0 & H_{2,m+1} & H_{2,m+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & H_{m,m} & H_{m,m+1} & H_{m,m+2} \\ H_{m+1,1} & H_{m+1,2} & \cdots & H_{m+1,m} & H_{m+1,m+1} & H_{m+1,m+2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \qquad (12)$$

Then the submatrix $\widehat{H}_{11}$, corresponding to the hyperlinks among the nondangling nodes, turns out to be

$$\widehat{H}_{11} = \begin{bmatrix} H_{11} & 0 & \cdots & 0 & H_{1,m+1} \\ 0 & H_{22} & \cdots & 0 & H_{2,m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & H_{m,m} & H_{m,m+1} \\ H_{m+1,1} & H_{m+1,2} & \cdots & H_{m+1,m} & H_{m+1,m+1} \end{bmatrix}. \qquad (13)$$

It is apparent that after the separation of the common nodes, the structure of the above matrix $\widehat{H}_{11}$ seems much simpler than the former one in (11).

3.3. A PageRank Algorithm. Notice that the matrix in (13) has nonzero submatrices only in the diagonal, the last row, and the last column. This special structure can reduce the computation in every iteration. Let

$$A = \begin{bmatrix} H_{11} & 0 & \cdots & 0 \\ 0 & H_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_{m,m} \end{bmatrix}, \qquad B = \begin{bmatrix} H_{1,m+1} \\ H_{2,m+1} \\ \vdots \\ H_{m,m+1} \end{bmatrix}, \tag{14}$$

$$E = \begin{bmatrix} H_{m+1,1} & H_{m+1,2} & \cdots & H_{m+1,m} \end{bmatrix},$$

$$F = \begin{bmatrix} H_{m+1,m+1} \end{bmatrix}.$$

Then

$$\widehat{H}_{11} = \begin{bmatrix} A & B \\ E & F \end{bmatrix}. \tag{15}$$

The coefficient matrix $(I - \alpha\widehat{H}_{11})$ has the following structure:

$$\left(I - \alpha\widehat{H}_{11}\right) = \begin{bmatrix} I - \alpha A & -\alpha B \\ -\alpha E & I - \alpha F \end{bmatrix}. \tag{16}$$

Therefore, after Gaussian elimination, $\widehat{x}_1^T(I - \alpha\widehat{H}_{11}) = \widehat{v}_1^T$ can be written as

$$y_1^T(I - \alpha A) = \alpha y_1^T B(I - \alpha F)^{-1}E + w_2^T(I - \alpha F)^{-1}E + w_1^T, \tag{17}$$

$$y_2^T = \left(w_2^T + y_1^T\alpha B\right)(I - \alpha F)^{-1}, \tag{18}$$

where $\widehat{x}_1^T = (y_1^T, y_2^T)$ and $\widehat{v}_1^T = (w_1^T, w_2^T)$ are divided into general and common sections. The only system that must be solved is (17).

Notice that the matrix A is a block diagonal matrix. Therefore, the subvectors x_i^T of $y_1^T = (x_1^T, x_2^T, \dots, x_3^T)$ which are partitioned according to the number and size of the blocks can be calculated independently in each iteration. For example, in kth iteration, calculate and divide $(w_2^T + y_1^{(k-1)T}\alpha B)(I - \alpha F)^{-1}E + w_1^T$ into $(r_1^T, r_2^T, \dots, r_m^T)$ according to the number and size of the blocksl then, for vectors x_i^T, we have the following function:

$$\left(x_1^{(k)T}, x_2^{(k)T}, \cdots, x_3^{(k)T}\right) \times \begin{bmatrix} I - \alpha H_{11} & 0 & \cdots & 0 \\ 0 & I - \alpha H_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I - \alpha H_{m,m} \end{bmatrix} = \left(r_1^T, r_2^T, \dots, r_m^T\right), \tag{19}$$

or

$$x_i^{(k)T}(I - \alpha H_{ii}) = r_i^T, \quad i = 1, 2, \dots, m. \tag{20}$$

As a result, the PageRank system in (8) can be reduced into the smaller linear system formulation in (17) in which the subvectors can be calculated independently in each iteration by (20). In summary, we now have an algorithm based on the separation of the common nodes. Meanwhile, this algorithm is an extension of the dangling node method in Section 2.

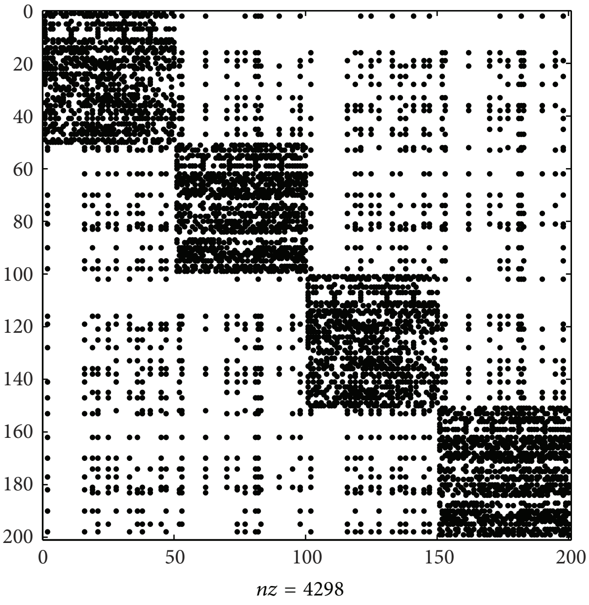


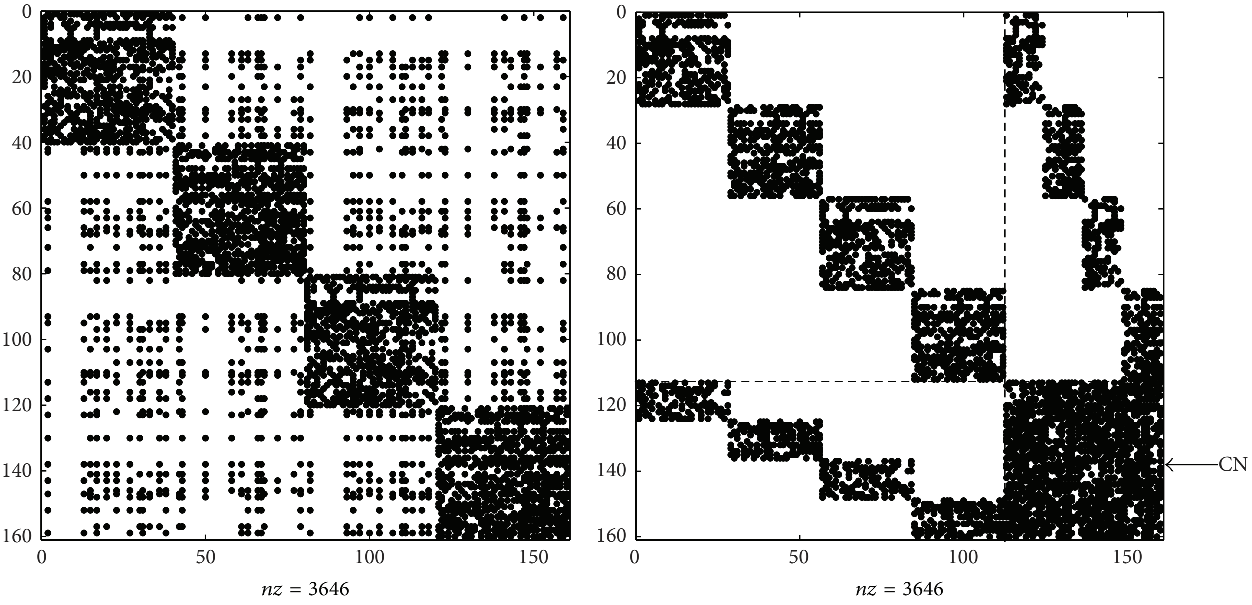
Figure 2: One of the three web link graphs, where the proportion between general nodes and common nodes is 7 : 3 in each subblock.

4. Analysis of Algorithm 2

As we know, some web link graphs appear to have a nested block structure. Then according to the definition of common node, it is not difficult to find the common nodes among the different blocks. This can be done by a process of locating nonzero entries on submatrices of $H_{i,j}$ in (10) $(i \neq j, 1 \leq i \leq m, 1 \leq j \leq m)$. For example, if the (k_1, k_2)th entry of $H_{i,j}$ is nonzero, then the k_1th nodes and the k_2th nodes are common nodes. This process can be repeated on different submatrices of $H_{i,j}$ at the same time by using separate computers. At the end, gather the common nodes together from different computers and get rid of the repetitive nodes, and then we get the last set of the common nodes. Since the dimension of $H_{i,j}$ is much smaller and we can use parallel searching, so the step 1 in Algorithm 2 will not take much time for separating the common nodes.

Note that there is no links among the new blocks $S_1^2, S_2^2, \dots, S_m^2$ after the separation of the common nodes just as the zero submatrices in the matrix H in (12). In effect, step 3 in Algorithm 2 reduces time consuming for large matrices by turning a large matrix $\widehat{H}_{11}$ into many smaller submatrices H_{ii}. It shows that vectors $x_i^{(k)T}$, $i = 1, \dots, m$, can be computed separately by $x_i^{(k)T}(I - \alpha H_{ii}) = r_i$ and the results are used together to yield a new vector for the next iteration. The parallel computation in this step can save much time.

Since x_i^T are not required to be accurate in each iteration, we can compute $x_i^{(k)T}$ by $x_i^{(k)T} = x_i^{(k-1)T}(\alpha H_{ii}) + r_i$. Moreover, it can be solved by any appropriate direct or iterative method. Meanwhile, in [22], they have found that acceleration methods [9, 11, 15, 26], such as extrapolation and preconditioners, can be applied to the small $H_{i,i}$ system to achieve even greater speedups.



(a) The web link graph of the submatrix $\widehat{H}_{11}$ before the reordering

(b) The web link graph of the submatrix $\widehat{H}_{11}$ after the reordering

Figure 3: A reordering of the submatrix $\widehat{H}_{11}$.

5. Numerical Experiments

5.1. Experiment Foundation. In this section, we give an example to present our algorithms.

Example. We consider three experiments based on three web link graphs: graph 1, graph 2, and graph 3. We assume that each of the graphs contains 200 nodes and four blocks; moreover, the size of the blocks is the same in each graph. Based on our definition about web pages, there are three classes of pages in a web: dangling nodes, common nodes, and general nodes. In order to make comparisons among the experiments, we suppose that the numbers of the dangling nodes are equivalent in these three graphs. Then we set different proportions of the general nodes and the common nodes in these three graphs. Without loss of generality, we assume that there are three kinds of proportions: they are 3 : 7 in graph 1, 5 : 5 in graph 2, and 7 : 3 in graph 3, which indicate that the number of the common nodes relatively decreases and the number of the general nodes relatively increases. We also assume that, in each graph, the proportion between the general nodes and the common nodes in each subblock is similar to the proportion in the whole web link graph. Meanwhile, in these three web link graphs, the choosing of the common nodes and the links in and between the subblocks is random.

For the dot plot graph of these three web link graphs, if there exists a link from node i to node j, then point (i, j) is colored; otherwise, point (i, j) is white. We assure that these three web link graphs satisfy three characters in [4].

(1) There is a definite block structure to the web.

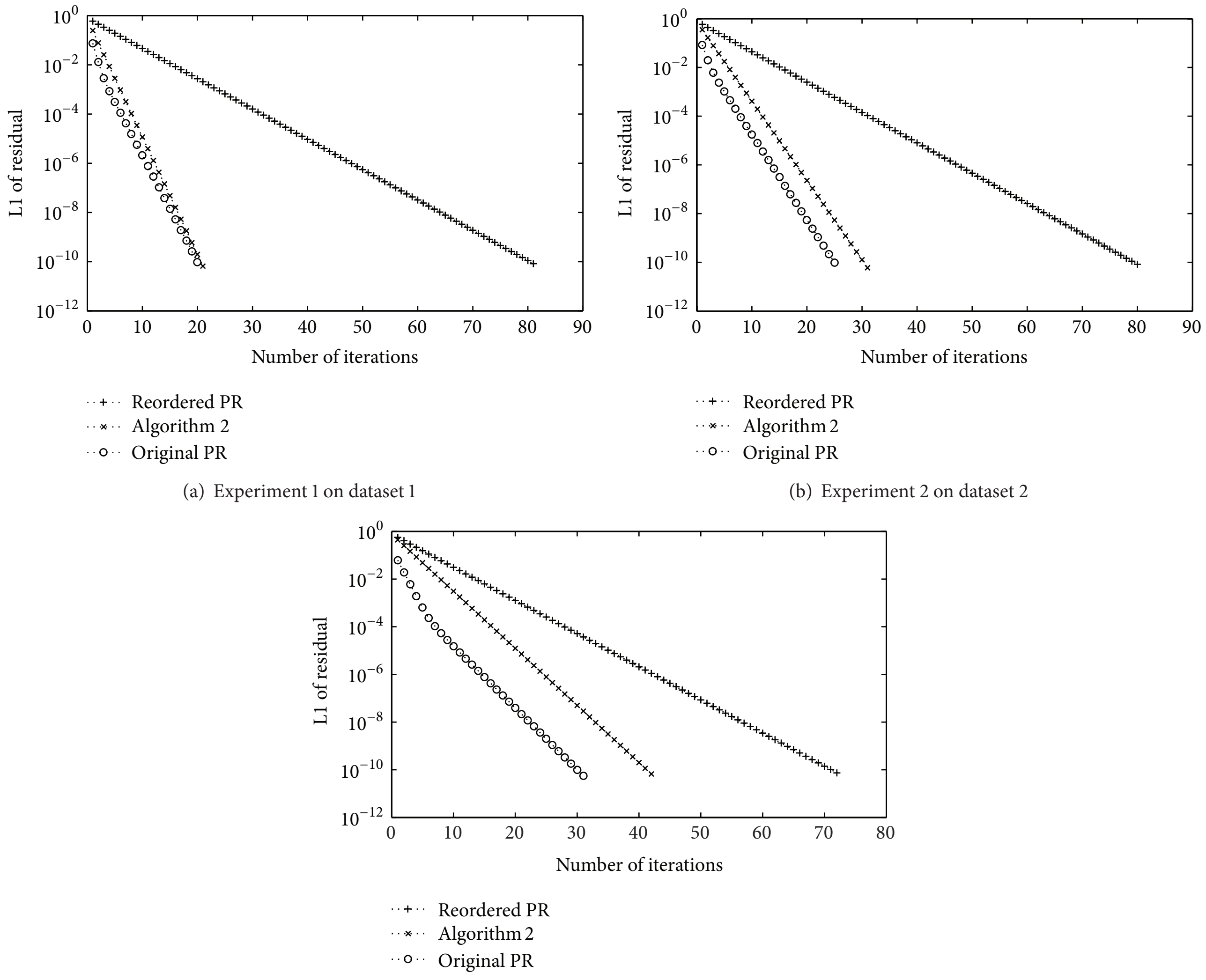
(2) The individual blocks are much smaller than entire web.

(3) There are clear nested blocks.

For example, Figure 2, it is the graph 3 which contains 200 pages and has a nested block structure of four blocks. The proportion is 7 : 3 in the whole graph.

Then, in each experiment, we separate the nodes into dangling nodes, common nodes, and the rest (general nodes). The result of this process is a decomposition of the H matrix. Figure 3 shows the change of the structure of $\widehat{H}_{11}$ in (4) after this process, which is based on the dataset of Figure 2. Figure 3(a) is the web link graph of $\widehat{H}_{11}$ before reordering, and Figure 3(b) is the new web link graph of $\widehat{H}_{11}$ after reordering. This process amounts to a simple reordering of the indices of the Markov chain. It shows that the character of the new structure is better than the original one.

5.2. Experimental Results and Analysis. Based on the three experiment datasets, we compare Algorithm 2 to the other two algorithms: original PageRank and reordered PageRank. We assume the scaling factor $\alpha = 0.85$ and the convergence tolerance $\tau = 10^{-10}$. The experimental results are shown in Figure 4 and Table 1. Figures 4(a), 4(b), and 4(c) are the comparison among the three algorithms about the acceleration of convergence in the three separate experiments. It shows that Algorithm 2 possesses both good capability to search PageRank vector and rigid convergence speed in comparison with reordered PageRank. That is because the dimension of the linear system for Algorithm 2 is smaller than the dimension of the linear system for reordered PageRank. The result in Table 1 implies that Algorithm 2 needs more iterations than Power method. However, since the application of parallel computation in Algorithm 2, Algorithm 2 can largely accelerate the computation time of PageRank. For the next work, we will try to experiment on real data.



(a) Experiment 1 on dataset 1

(b) Experiment 2 on dataset 2

(c) Experiment 3 on dataset 3

Figure 4: Comparison among the three algorithms which are run on three datasets.

Table 1: Comparison of original PageRank, reordered PageRank and Algorithm 2.

		Dataset 1	Dataset 2	Dataset 3
Reordered PageRank	Iterations	81	80	72
	Time (sec.)	0.0377	0.0366	0.0400
Original PageRank	Iterations	20	25	31
	Time (sec.)	0.0292	0.0270	0.0305
Algorithm 2	Iterations	21	31	42
	Time (sec.)	0.0165	0.0145	0.0187

6. Conclusion

It has investigated that the hyperlink graphs of some web pages have nested block structure which can be found in [4]. Then we exploit a reordered block structure and present an algorithm to compute PageRank in a fast manner. Algorithm 2 has basically two stages. In Stage 1, the focus is on the partition of nodes in a web. In Stage 2, the vector of general nodes in each block for next iteration is computed independently. Then we calculate the unnormalized PageRank vectors for common nodes and dangling nodes directly. At last, normalize the vector and give the PageRank. The numerical experiments show that Algorithm 2 is guaranteed to outperform the other two algorithms, as long as an appropriate block structure of web exists. However, in real data, the common nodes may increase as the number of the blocks increases, and the dimension of the submatrix D could be larger. Then it will take much time to calculate the value of $(I - \alpha D)^{-1}$. In this case, similar to Algorithm 2, we will consider calculating the vector for common nodes first and then calculating the vector for general nodes in each block independently. We aslo need to experiment on real data and make comparison with other more existing methods in the future work.

Acknowledgments

The authors would like to express their great thankfulness to the referees and the editor for their much helpful suggestions for revising this paper. This research is supported by 973 Program (2013CB329404), NSFC (61370147, 61170311, and 61170309), Chinese Universities Specialized Research Fund for the Doctoral Program (20110185110020), and Sichuan Province Sci. & Tech. Research Project (2012GZX0080).

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] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[3] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (SALSA) and the TKC effect," in *Proceedings of the 9th International Conference on the World Wide Web*, pp. 387–401, ACM Press, New York, NY, USA, 2000.

[4] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Exploiting the block structure of the web for computing PageRank," Tech. Rep. 2003-17, Stanford University, Stanford, Calif, USA, 2003.

[5] E. Minkov, "Adaptive graph walk based similarity measures in entity-relation graphs," type CMU-LTI-09-004, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pa, USA, 2008.

[6] T. H. Haveliwala and S. D. Kamvar, "The second eigenvalue of the Google matrix," Tech. Rep. 2003-20, Stanford University, Stanford, Calif, USA, 2003.

[7] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, Third edition, 1996.

[8] R. S. Wills and I. C. F. Ipsen, "Ordinal ranking for Google's PageRank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 4, pp. 1677–1696, 2008/09.

[9] 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*, ACM Press, New York, NY, USA, 2002.

[10] A. Broder, R. Kumar, and F. Maghoul, "Graph structure in the web: experiments and models," in *Proceedings of the 9th International World Wide Web Conference*, 2000.

[11] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolation methods for accelerating the computation of PageRank," in *Proceedings of the 12th International World Wide Web Conference*, pp. 261–270, ACM Press, New York, NY, USA, 2003.

[12] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte Carlo methods in pagerank computation: when one iteration is sufficient," *SIAM Journal on Numerical Analysis*, vol. 45, no. 2, pp. 890–904, 2007.

[13] D. F. Gleich, A. P. Gray, C. Greif, and T. Lau, "An inner-outer iteration for computing PageRank," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 349–371, 2010.

[14] C. P. Lee, G. H. Golub, and S. A. Zenios, "Partial state space aggregation based on lumpability and its application to PageRank," Tech. Rep., Stanford University, 2003.

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

[16] A. Cevahir, C. Aykanat, A. Turk, and B. B. Cambazoglu, "Site-based partitioning and repartitioning techniques for parallel pagerank computation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 786–802, 2011.

[17] D. Gleich, L. Zhukov, and P. Berkhin, "Fast parallel PageRank: a linear system approach," Tech. Rep. YRL-2004-038, 2004.

[18] C. Kohlschtter, R. Chirita, and W. Nejdl, "Efficient parallel computation of PageRank," in *Proceedings of the 28th European Conference on IR Research (ECIR '06)*, pp. 241–252, 2006.

[19] G. Kollias and E. Gallopoulos, "Asynchronous PageRank computation in an interactive multithreading environment," in *Proceedings of the Seminar Web Information Retrieval and Linear Algebra Algorithms*, 2007.

[20] G. Kollias, E. Gallopoulos, and D. B. Szyld, "Asynchronous iterative computations with web information retrieval structures: the PageRank case," in *Proceedings of the ParCo*, 2006.

[21] B. Manaskasemsak and A. Rungsawang, "An efficient partition-based parallel PageRank algorithm," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems Workshops (ICPADS '05)*, pp. 257–263, July 2005.

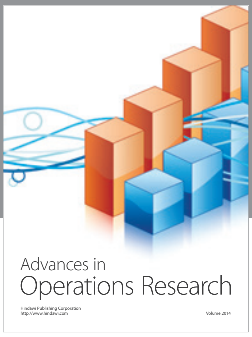
[22] A. N. Langville and C. D. Meyer, "A reordering for the PageRank problem," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 2112–2120, 2006.

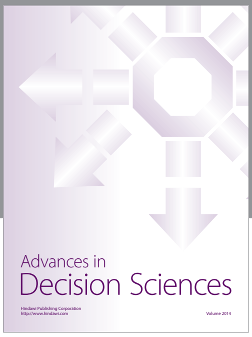
[23] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, vol. 9, SIAM, Philadelphia, Pa, USA, 1994.

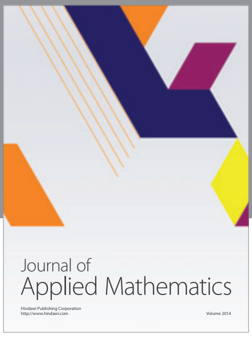
[24] C. P. Lee, G. H. Golub, and S. A. Zenios, "A fast two-stage algorithm for computing PageRank and its extensions," Tech. Rep. SCCM-03-15, Stanford University, Stanford, Calif, usa, 2003.

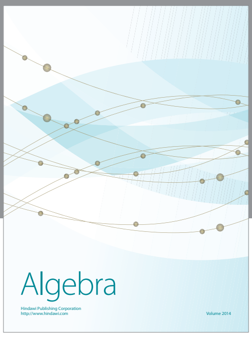
[25] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke, "WebBase: a repository of Web pages," *Computer Networks*, vol. 33, no. 1, pp. 277–293, 2000.

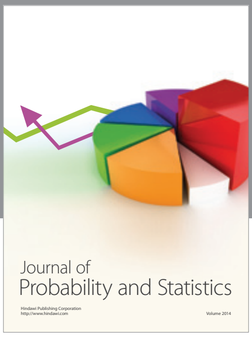
[26] G. H. Golub and C. Greif, "Arnoldi-type algorithms for computing stationary distribution vectors, with application to PageRank," Tech. Rep. SCCM-2004-15, Scientific Computation and Computational Mathematics, Stanford University, Stanford, Calif, USA, 2004.


Advances in
Operations Research

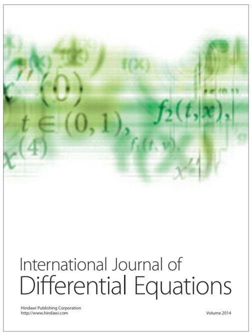

Advances in
Decision Sciences


Journal of
Applied Mathematics

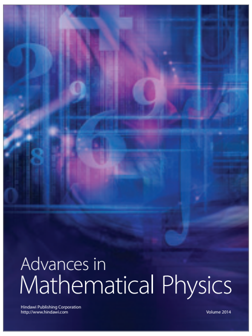

Algebra

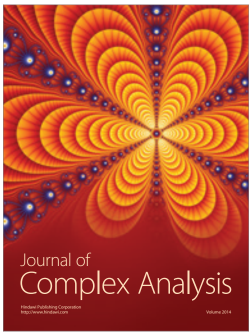

Journal of
Probability and Statistics

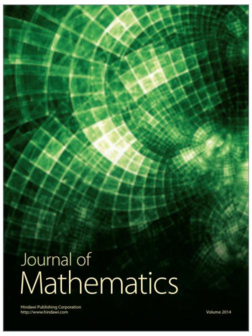
The Scientific
World Journal

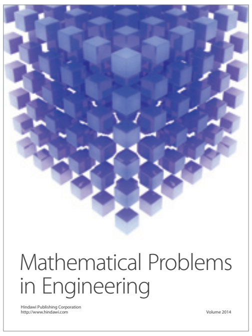

International Journal of
Differential Equations

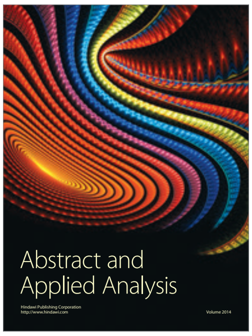
International Journal of
Combinatorics

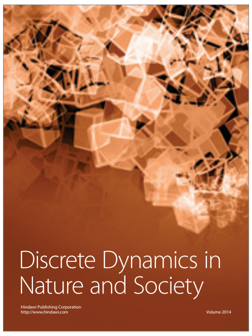

Advances in
Mathematical Physics

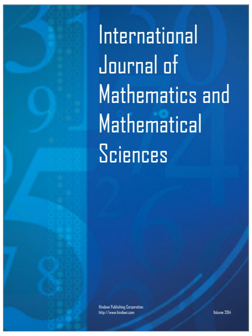

Journal of
Complex Analysis


Journal of
Mathematics


Mathematical Problems
in Engineering

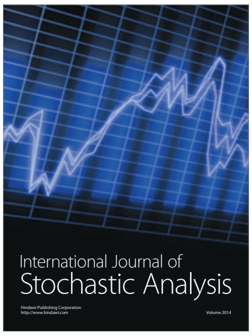

Abstract and
Applied Analysis


Discrete Dynamics in
Nature and Society


International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces


International Journal of
Stochastic Analysis

Journal of
Optimization