Data Min Knowl Disc (2013) 26:27–56
DOI 10.1007/s10618-011-0245-7

A preconditioned conjugate gradient algorithm for GeneRank with application to microarray data mining

Gang Wu · Wei Xu · Ying Zhang · Yimin Wei

Received: 14 February 2011 / Accepted: 27 October 2011 / Published online: 13 November 2011

Abstract The problem of identifying key genes is of fundamental importance in biology and medicine. The GeneRank model explores connectivity data to produce a prioritization of the genes in a microarray experiment that is less susceptible to variation caused by experimental noise than the one based on expression levels alone. The GeneRank algorithm amounts to solving an unsymmetric linear system. However, when the matrix in question is very large, the GeneRank algorithm is inefficient and even can be infeasible. On the other hand, the adjacency matrix is symmetric in the GeneRank model, while the original GeneRank algorithm fails to exploit the symmetric structure of the problem in question. In this paper, we discover that the GeneRank problem can be rewritten as a symmetric positive definite linear

Responsible editor: Pierre Baldi.

Dedicated to Prof. Zhi-hao Cao on the occasion of his 75th birthday.

G. Wu
School of Mathematical Sciences, Xuzhou Normal University, Xuzhou 221116, Jiangsu, People's Republic of China
e-mail: gangwu76@yahoo.com.cn; wugangzy@gmail.com

W. Xu (✉)
Department of Mathematics, Tongji University, Shanghai 200092, People's Republic of China
e-mail: wdxu@tongji.edu.cn

Y. Zhang
Xuzhou Medical College, Xuzhou 221000, Jiangsu, People's Republic of China
e-mail: drzhangying@yahoo.com.cn

Y. Wei
Shanghai Key Laboratory of Contemporary Applied Mathematics, School of Mathematical Sciences, Fudan University, Shanghai 200433, People's Republic of China
e-mail: ymwei@fudan.edu.cn; yimin.wei@gmail.com

system, and propose a preconditioned conjugate gradient algorithm to solve it. Numerical experiments support our theoretical results, and show superiority of the novel algorithm.

Keywords Gene network · Microarray · GeneRank · Preconditioned conjugate gradient method (PCG) · Krylov subspace method

1 Introduction

The problem of identifying key genes that are involved in a particular disease is of fundamental importance in biology and medicine. Given the increasing availability of a variety of gene-related biological data sources, ranging from microarray expression data to protein–protein interaction (PPI) data, a promising approach is to use bioinformatics methods that can analyze this data and rank genes based on potential relevance to a disease; such methods can be invaluable in helping to prioritize genes for further biological study. However, how to prioritize genes for phenotypes when no candidates are available is still a challenging problem (Agarwal and Sengupta 2009; Ma et al. 2007).

Prioritizing genes based on the accumulated information from several data sources has attracted much attention recently. For example, Franke et al. (2006) proposed to rank genes in candidate regions by their connectivity with the genes in other linked regions. However, this method depends strongly on the availability of results from genetics studies. Aerts et al. (2006) developed a Bayesian model to identify new genes, in addition to the already known genes, involved in a given disease, using many currently available data types. However, they assumed that a set of genes responsible for the disease is already known, which limits the applicability of their method (Ma et al. 2007). Agarwal and Sengupta (2009) rewrote the gene ranking problem to particular form of ranking problem termed the bipartite ranking problem. Recently, Ma et al. (2007) developed a new method for prioritizing genes associated with a phenotype by combining Gene expression and protein interaction data (CGI). The method is applied to yeast gene expression data sets in combination with protein interaction data sets of varying reliability.

Gene expression data and protein interaction data have been integrated for gene function prediction. A microarray is a sequence of dots of DNA, protein, or tissue arranged on an array for easy simultaneous experiments. The most famous one is the DNA microarray, which plays an important role in gene expression profiling. Interpretation of simple microarray experiments is usually based on the fold-change of gene expression between a reference and a "treated" sample, where the treatment can be of many types from drug exposure to genetic variation. Interpretation of the results usually combines lists of differentially expressed genes with previous knowledge about their biological function. However, some existing methods are heuristic and empirical, and are not efficient enough in many cases (Morrison et al. 2005). A new mathematical model, called *GeneRank*, was proposed by Morrison et al. in 2005. In essence, GeneRank is an intuitive modification of Google's PageRank (Page et al. 1998) that maintains many of its mathematical properties. The aim of the GeneRank model is to use connectivity data to produce a prioritization of the genes in a microarray experiment which is less susceptible to variation caused by experimental noise than one based on expression levels alone (Morrison et al. 2005). This model gives rise to a large sparse system of linear equations, which must be solved in practice.

A gene network can be constructed naturally via the GO database (http://geneontology.org). Let the set $G = \{g_1, g_2, \ldots, g_n\}$ be n genes in a microarray. Similar to the idea of PageRank (Page et al. 1998), if a gene is connected with many highly ranked genes, it should be highly ranked as well, even if it may have a low rank from the experimental data (Morrison et al. 2005). If two genes share at least one annotation in GO, they are defined to be connected. From this idea, we can build a gene network whose adjacent matrix is W, with elements

$$w_{ij} = \begin{cases} 1, & \text{if } g_i \text{ and } g_j \ (i \neq j) \text{ have the same} \\ & \text{annotation in GO,} \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

Indeed, the adjacent matrix W is symmetric since the network is undirected, i.e., $W^T = W$. This is unlike the PageRank problem, where a directed network is constructed. If we define

$$deg_i = \sum_{j=1}^{n} w_{ij} = \sum_{j=1}^{n} w_{ji} \tag{1.2}$$

to be the out-degree of the i-th gene, and denote by the diagonal matrix $D = diag(d_1, d_2, \ldots, d_n)$, where for $i = 1, 2, \ldots, n$,

$$d_i = \begin{cases} deg_i, & \text{if the } i\text{-th row of } W \text{ is nonzero,} \\ 1, & \text{otherwise.} \end{cases} \tag{1.3}$$

Then the GeneRank problem resorts to the following large scale unsymmetric linear system (Morrison et al. 2005)

$$(I - \alpha \cdot WD^{-1})x^* = (1 - \alpha) \cdot \mathbf{ex}, \tag{1.4}$$

where α is the *damping factor* with $0 < \alpha < 1$, and $\mathbf{ex} = [ex_1, ex_2, \ldots, ex_n]^T$, with $ex_i \geq 0$, is the absolute value of the expression change for g_i, $i = 1, 2, \ldots, n$. The solution x^* is called the GeneRank vector, whose elements can reflect the importance of the corresponding genes in some degree (Morrison et al. 2005).

We point out that WD^{-1} is a diagonalizable matrix. Consequently, it has a complete set of eigenvectors. Indeed, it is similar to the symmetric matrix as follows

$$D^{-1/2}(WD^{-1})D^{1/2} = D^{-1/2}WD^{-1/2}, \tag{1.5}$$

however, WD^{-1} is unsymmetric in general, and it is not guaranteed that the matrix is unitary diagonalizable.

Morrison et al. (2005) proposed a GeneRank algorithm for computing GeneRank. The MATLAB (The MATHWORKS, Inc. 2004) implementation of this algorithm is listed as follows.

Algorithm 1 [1] **The GeneRank algorithm (GeneRank)**

```
function r = GeneRank(W,ex,a)
$ex = abs(ex);$
$normex = \mathbf{ex}/max(ex);$
$w = sparse(W);$
$degrees = sum(W);$
$ind = find(degrees == 0);$
$degrees(ind) = 1;$
$D_1 = sparse(diag(1./degrees));$
$A = eye(size(w)) - a * (w' * D_1);$
$b = (1 - a) * ex;$
$r = A \setminus b;$
```

Note that W is a sparse matrix in general. However, the GeneRank algorithm uses a direct method to solve (1.4). This is *impractical* when n, the dimension of the matrix is very large; and the GeneRank algorithm may not be effective and even may fail to work in this case (Wu et al. 2010). On the other hand, due to a great amount of gene expression data required, efficient algorithms are essential (Hai et al. 2008). Recently, Yue et al. (2007) reformulated the GeneRank model as a linear combination of three parts, and presented an explicit formulation for the GeneRank vector. In Wu et al. (2010), the GeneRank problem was rewritten as a large scale eigenvalue problem, and it was solved by some Arnoldi-type algorithms.

Although the adjacent matrix W is *symmetric*, to our best knowledge, all the known algorithms resort the computation of GeneRank to *unsymmetric* matrix computation problems, refer to Sect. 5.1 in the Appendix. Therefore, it is necessary to seek a novel algorithm that exploits the symmetric structure of the GeneRank matrix. In this paper, we point out that the unsymmetric linear system (1.4) can be rewritten as a symmetric positive definite (SPD) linear system, and we solve it using a preconditioned conjugate gradient (PCG) method with a well-determined preconditioner. Finally, the numerical experiments made in Sect. 3 support our theoretical results, and illustrate superiority of the new algorithm for GeneRank. MATLAB files of the PCG algorithm and the modified Arnoldi algorithm for GeneRank (Wu et al. 2010) are provided in the Appendix.

2 A PCG algorithm for GeneRank

Efficiently numerical methods are essential for microarray data mining (Hai et al. 2008), because a great amount of gene expression data are required in practice. It is now one of the hot topics in molecular biology and brings new challenges for seeking efficient algorithms for very large gene networks.

Due to some constraints, biological, physical, and social sciences problems often have some special structures. In the GeneRank model, the adjacent matrix W is a

[1] http://www.biomedcentral.com/content/supplementary/1471-2105-6-233-S1.m. Accessed 2010.

symmetric matrix. To our best knowledge, all the known algorithms resort the GeneRank problem to solve an unsymmetric linear system, or to solve an unsymmetric eigenvalue problem. In other words, none of the known algorithms explore the structure of W efficiently. So the question is: Can we transform the GeneRank problem to a *symmetric* matrix computation problem, such that more efficient numerical algorithms can be applied? In this section, we first rewrite the unsymmetric linear system (1.4) as a SPD linear system. Then, we propose a strategy to determine an efficient preconditioner for the new system. Also, some theoretical properties are exploited.

2.1 A SPD linear system for GeneRank

In this subsection, we rewrite the GeneRank problem to a SPD linear system, and explore some properties of the coefficient matrix. The key idea stems from the fact that the Eq. 1.4 can be rewritten as

$$(D - \alpha W) \cdot \hat{x} = (1 - \alpha) \cdot \mathbf{ex}, \tag{2.1}$$

where $\hat{x} = D^{-1}x^*$. Thus, we can obtain the GeneRank vector from solving (2.1). It is easy to see that $D - \alpha W$ is a structured matrix with non-positive off-diagonal and nonnegative diagonal entries. Indeed, some nice structures and properties of $D - \alpha W$ can be further investigated. The following theorem shows that it is a SPD matrix.

Theorem 1 *$D - \alpha W$ is a symmetric positive definite* (*SPD*) matrix.

Proof See the Appendix. □

Since $D - \alpha W$ is SPD, Theorem 1 indicates that we can solve (2.1) using the famous conjugate gradient (CG) method (Hestenes and Stiefel 1952), which is ranked among *the Top 10 Algorithms* of the twentieth century (Cipra 2000). The CG method proposed by Hestenes and Stiefel in (1952), is one of the best known iterative methods for solving a SPD linear system $Ax = b$. The method is a realization of an orthogonal projection technique onto the Krylov subspace $\mathcal{K}_m(A, r_0)$, where $r_0 = b - Ax_0$ is the initial residual. The CG method is the "best" method in the sense that the approximation x_k minimizes $||x_k - x||_A$, with $||v||_A \equiv (v^T Av)^{1/2}$. Another merit of the CG method is that, in exact arithmetic, one can obtain the exact solution of an n by n linear system at most n steps (Demmel 1997; Golub and Loan 1996; Saad 2003). The prototype of the CG algorithm for GeneRank is listed as follows.

Algorithm 2 The conjugate gradient algorithm for GeneRank (CG)
Given a prescribed tolerance tol, and set $k = 0,\ x_0 = \mathbf{0},\ r_0 = \mathbf{ex},\ p_1 = \mathbf{ex},\ \mathbf{d} = [d_1, d_2, \ldots, d_n]^T$.

```
while $||r_k||_1 > tol$
    $k = k + 1;$
    % Here ".*" is the entry multiplication in MATLAB
    $z_k = \mathbf{d} .* p_k - \alpha * (W * p_k);$
    $\nu_k = (r_{k-1}^T r_{k-1})/(p_k^T z);$
    $x_k = x_{k-1} + \nu_k p_k;$
    $r_k = r_{k-1} - \nu_k z_k;$
    $\mu_{k+1} = (r_k^T r_k)/(r_{k-1}^T r_{k-1});$
    $p_{k+1} = r_k + \mu_{k+1} p_k;$
end
```

In Algorithm 2, each iteration requires performing a matrix-vector product, and storing five vectors $\mathbf{d}, x, p, z$ and r. The following theorem shows convergence of the CG method (Demmel 1997; Golub and Loan 1996; Saad 2003) applied to the GeneRank problem.

Theorem 2 *Denote by $\widetilde{D} \equiv D - \alpha W$, and by $\hat{x}$ the exact solution of Eq. 2.1. Then*

$$||\hat{x} - x^{(k)}||_{\widetilde{D}} \leq 2\left(\frac{\sqrt{\kappa_2(D-\alpha W)} - 1}{\sqrt{\kappa_2(D-\alpha W)} + 1}\right)^k ||\hat{x} - x^{(0)}||_{\widetilde{D}}, \tag{2.2}$$

where $x^{(k)}$ is produced by the CG method,

$$\kappa_2(D - \alpha W) = \frac{\lambda_{\max}(D - \alpha W)}{\lambda_{\min}(D - \alpha W)} \tag{2.3}$$

is the 2-norm condition number of $\widetilde{D}$, and $||v||_{\widetilde{D}}^2 \equiv v^T(D - \alpha W)v$.

Theorem 2 indicates that, the number of CG iterations needed to reduce the error by a fixed factor less than 1, is proportional to the square root of the two-condition number $\kappa_2(D - \alpha W)$. Here the two-condition number is the ratio of the largest eigenvalue to the smallest eigenvalue of $D - \alpha W$. Indeed, the convergence rate is determined not only by the ratio of the largest one to the smallest one, but also by the entire distribution of the eigenvalues (Demmel 1997). When the eigenvalues are clustered together, the CG method will converge much faster than the theoretical analysis based on $D - \alpha W$'s condition number. On the other hand, if the eigenvalues of $D - \alpha W$ are not distributed well, then the CG method may converge slowly. One remedy for this situation is to use the preconditioning techniques (Saad 2003).

2.2 Preconditioner determination

As we know that the convergence rate of the CG method depends on the condition number of the matrix in question, or more generally the distribution of eigenvalues. If the eigenvalue distribution of the preconditioned system is better than that of the original one, the convergence will be accelerated dramatically. In other words, a preconditioner is any form of implicit or explicit modification of an original linear system, that makes it easier to be solved by a given iterative method. In fact, a good preconditioner is often necessary for an iterative method to converge, and much current research in iterative methods is directed at finding better preconditioners (Benzi 2002; Saad 2003).

Given a linear system $Ax = b$, preconditioning means replacing the original system with the system $M^{-1}Ax = M^{-1}b$, where M is called the *preconditioner*. For a SPD linear system, a good preconditioner is an approximation to A with the properties as follows (Demmel 1997):

(1) M is symmetric and positive definite.
(2) $M^{-1}A$ is well conditioned or has few extreme eigenvalues.
(3) $Mx = b$ is easy to solve.

However, we can not apply CG directly to the system $M^{-1}Ax = M^{-1}b$, since $M^{-1}A$ is unsymmetric in general. Indeed, we can precondition the system as follows

$$(M^{-1/2}AM^{-1/2})(M^{1/2}x) = M^{-1/2}b, \tag{2.4}$$

where $M^{1/2}$ is the square root of the SPD matrix M. Note that $M^{-1}A$ and $M^{-1/2}AM^{-1/2}$ share the same eigenvalues. In this paper, we propose to use the diagonal matrix

$$M = D = diag(d_1, d_2, \ldots, d_n), \tag{2.5}$$

as a preconditioner for the linear system (2.1), where the d_i's are defined in (1.3). It follows that the preconditioned linear system becomes

$$(I - \alpha \cdot D^{-1/2}WD^{-1/2})(D^{1/2}\hat{x}) = (1-\alpha)(D^{-1/2}\mathbf{ex}). \tag{2.6}$$

The following theorem reveals the spectral property of the new coefficient matrix $I - \alpha \cdot D^{-1/2}WD^{-1/2}$, which also demonstrates the rationality of our preconditioner.

Theorem 3 *The largest and the smallest eigenvalue of $I - \alpha \cdot D^{-1/2}WD^{-1/2}$ satisfies*

$$\lambda_{\max}(I - \alpha \cdot D^{-1/2}WD^{-1/2}) \leq 1 + \alpha, \tag{2.7}$$

and

$$\lambda_{\min}(I - \alpha \cdot D^{-1/2}WD^{-1/2}) \geq 1 - \alpha, \tag{2.8}$$

respectively.

Proof See the Appendix. □

Remark 1 We point out that the preconditioner D satisfies all the three conditions described above. Firstly, note that D is a positive diagonal matrix, so it is symmetric and positive definite. Another merit of this choice is that we only need to store the preconditioner using an n-dimensional vector. Secondly, the linear system $Dx = b$ is easy to solve, i.e., $x = D^{-1}b$, with $\mathcal{O}(n)$ flops. Finally, Theorem 3 shows that eigenvalues of the preconditioned matrix $I - \alpha \cdot D^{-1/2}WD^{-1/2}$ are clustered around 1 when α is medium size. For instance, if we take $\alpha = 0.85$, the damping factor used by Google (Page et al. 1998), then $\lambda_{\max}(I - \alpha \cdot D^{-1/2}WD^{-1/2}) \leq 1.85$, and $\lambda_{\min}(I - \alpha \cdot D^{-1/2}WD^{-1/2}) \geq 0.15$.

However, in practical calculations, it is unnecessary to apply the CG method to the preconditioned system (2.4) *directly*. The reason it that we have to multiply $M^{-1/2}$ twice per iteration. An equivalent alternative is to apply CG *implicitly* to the system (2.4), in such a way that avoids multiplying by $M^{-1/2}$ (Demmel 1997; Golub and Loan 1996; Saad 2003). The main algorithm of this paper is outlined as follows.

Algorithm 3 A preconditioned conjugate gradient algorithm for GeneRank (PCG)
Given a prescribed tolerance tol, the damping factor α*, and* $\mathbf{d} = [d_1, d_2, \ldots, d_n]^T$*; set* $k = 0$, $x_0 = \mathbf{O}$, $r_0 = \mathbf{ex}$, $p_1 = \mathbf{ex}./\mathbf{d}$, $y_0 = r_0./\mathbf{d}$*;*

```
while $||r_k||_1 > tol$
    $k = k + 1;$
    $z_k = \mathbf{d}.*p_k - \alpha * (W * p_k);$
    $\nu_k = (y_{k-1}^T r_{k-1})/(p_k^T z_k);$
    $x_k = x_{k-1} + \nu_k p_k;$
    $r_k = r_{k-1} - \nu_k z_k;$
    $y_k = r_k./\mathbf{d};$
    $\mu_{k+1} = (y_k^T r_k)/(y_{k-1}^T r_{k-1});$
    $p_{k+1} = y_k + \mu_{k+1} p_k;$
end
$x^* = \mathbf{d}.*x_k;$    % the GeneRank vector
```

Remark 2 In Algorithm 3, we apply the CG method *implicitly* to the system (2.6) (or in other words, apply the PCG method to the system (2.1)), without multiplying by $D^{-1/2}$. In the PCG algorithm, it needs to store six vectors of length n, and to perform one matrix-vector multiplication in each iteration.

The following result shows convergence of the PCG method for GeneRank.

Corollary 1 *Denote by* $\hat{x}^* = D^{1/2}\hat{x} = D^{-1/2}x^*$*, and by* $x^{(k)}$ *the approximation obtained from the k-th iteration of the PCG method applies to the system* (*2.1*)*. Then under the above notation, we have*

$$||\hat{x}^* - x^{(k)}||_{\widehat{D}} \leq 2\left(\frac{\sqrt{1+\alpha} - \sqrt{1-\alpha}}{\sqrt{1+\alpha} + \sqrt{1-\alpha}}\right)^k ||\hat{x}^* - x^{(0)}||_{\widehat{D}}, \tag{2.9}$$

where $\widehat{D} \equiv I - \alpha \cdot D^{-1/2} W D^{-1/2}$, *and* $||v||_{\widehat{D}}^2 \equiv v^T(I - \alpha \cdot D^{-1/2} W D^{-1/2})v$.

Proof See the Appendix. □

An interesting question is how correlated are the final GeneRank vectors obtained from the different methods. To this aim, we give insight into sine of the angles between the approximate vectors. Notice that the smaller the sine of the angles, the more correlated the approximations. Let x_1 and x_2 be the solutions from two different methods, and x^* be the exact GeneRank vector. Since

$$\sin\angle(x_1, x_2) \leq \sin\angle(x_1, x^*) + \sin\angle(x_2, x^*). \tag{2.10}$$

Thus, it is sufficient to consider the angles between the approximate solutions and the exact solution x^*. The following theorem reveals that, for the algorithms discussed in this paper (see the Appendix), the angles between the approximate solutions and x^* will approach zero as the corresponding residuals do, provided that the damping factor is not too close to 1.

Theorem 4 (i) *For the Jacobi iteration, we have*

$$\sin \angle(x^{(k)}, x^*) \leq \alpha^k \cdot \sqrt{n}||x^{(0)} - x^*||_2, \tag{2.11}$$

where $x^{(0)}$ is the initial vector for the Jacobi iteration and n is the size of the matrix in question.

(ii) *Let $X_\perp$ be an orthonormal basis for the orthogonal complement of the space spanned by the GeneRank vector x^*, so that $[x^*,\ X_\perp]$ is orthonormal, then we have*

$$\begin{bmatrix} (x^*)^T \\ X_\perp^T \end{bmatrix} G \begin{bmatrix} x^* & X_\perp \end{bmatrix} = \begin{bmatrix} 1 & (x^*)^T A X_\perp \\ \mathbf{O} & G_2 \end{bmatrix}. \tag{2.12}$$

Denote by ϵ^G, ϵ^M the distance between the GeneRank vector x^* and the search subspaces of the Arnoldi algorithm and the modified Arnoldi algorithm, respectively. Then, for the Arnoldi algorithm

$$\sin \angle(x^G, x^*) \leq \frac{\epsilon^G}{\sqrt{1-(\epsilon^G)^2}} \cdot ||G - I||_2 \left\| (I - G_2)^{-1} \right\|_2, \tag{2.13}$$

and for the modified Arnoldi algorithm, we have

$$\sin \angle(x^M, x^*) \leq \frac{\epsilon^M}{\sqrt{1-(\epsilon^M)^2}} \cdot ||G - I||_2 \left\| (I - G_2)^{-1} \right\|_2. \tag{2.14}$$

(iii) *Under the notation of Theorem 2 and Corollary 1, we have*

$$\sin \angle(\hat{x}, x^{(k)}) \leq \frac{1}{\sqrt{\lambda_{\min}(\widetilde{D})}} ||\hat{x} - x^{(k)}||_{\widetilde{D}}. \tag{2.15}$$

for the CG method, and

$$\sin \angle(\hat{x}^*, x^{(k)}) \leq \frac{1}{\sqrt{1-\alpha}} ||\hat{x}^* - x^{(k)}||_{\widehat{D}}. \tag{2.16}$$

for the PCG method, respectively.

Proof See the Appendix. □

3 Numerical examples

In this section, we try to make some numerical experiments to illustrate numerical behavior of the new algorithm. One is recommended to see the Appendix for some iterative algorithms for GeneRank, including the Jacobi iteration (Jacobi), the Arnoldi algorithm (Arnoldi) (Wu et al. 2010), as well as the modified Arnoldi algorithm (Modi-Arn) (Wu et al. 2010). All the numerical experiments were run on a dual core Intel(R) Pentium(R) processor with CPU 1.86 GHz and RAM 1GB under the Windows XP operating system. All the numerical results were obtained from a MATLAB 7.0 implementation with machine precision $\epsilon \approx 2.22 \times 10^{-16}$. It is pointed out that the optimal choice of α is data-dependent (Morrison et al. 2005), and the choice of the damping factor is crucial to the GeneRank model. In this section, we choose $\alpha = 0.5,\ 0.7,\ 0.85$ and 0.99, respectively.

In Examples 1 and 3, there are three adjacency matrix w_All, w_Up and w_Down, and three expression change vectors $expr\_data$, $expr\_dataUp$ and $expr\_dataDown$. These matrices were constructed using the all three sections of the Gene Ontology, where a link is presented between two genes if they share a GO annotation. Only genes which are up-regulated are included in w_Up and only down-regulated in w_Down (Morrison et al. 2005). The data files are available from.[2]

Example 1 In this example, we try to show efficiency of Algorithm 3 (PCG) for computing Gene-Rank. We compare PCG with the GeneRank algorithm (Morrison et al. 2005) and the Jacobi iteration, where the initial guess for the two iterative algorithms is both set to be zero. The first test matrix is the w_All matrix, it is of size $4,047$ by $4,047$, with $339,596$ nonzero elements, and $\mathbf{ex} = extr\_data$; the second test matrix is the w_Up matrix, which is of size $2,392 \times 2,392$, with $128,468$ nonzero elements, and $\mathbf{ex} = extr\_dataUp$.

We notice from Algorithm 1 that the GeneRank algorithm exploits a direct method for the linear system (1.4). For the sake of justification, the stopping criterion for the PCG method is

$$||r_k||_1 \leq tol, \tag{3.1}$$

where r_k is the residual of the k-th iteration; and the stopping criterion for the Jacobi iteration is

$$\|x^{(k)} - x^{(k-1)}\|_1 \leq tol, \tag{3.2}$$

where $x^{(k)}$ is the approximation from the k-th iteration, and $tol = 10^{-14}$.

Tables 1 and 2 list the number of matrix-vector multiplications and the CPU time in seconds (in brackets) for the three algorithms on these two testing problems. We observe that both the two iterative algorithms run much faster than the GeneRank algorithm, while the PCG method is superior to the Jacobi iteration both in terms of

[2] http://www.biomedcentral.com/content/supplementary/1471-2105-6-233-S2.mat. Accessed 2010.

Table 1 The w_All matrix, $n = 4,047$, $tol = 10^{-14}$

α	0.50	0.70	0.85	0.99
Jacobi	39 (0.30)	74 (0.55)	162 (1.22)	2604 (20.1)
PCG	23 (0.22)	31 (0.27)	42 (0.36)	67 (0.59)
GeneRank	(31.0)	(31.2)	(31.0)	(31.1)

Example 1: the number of matrix-vector products and the CPU time in seconds (in *brackets*) of the three algorithms. We see that PCG works better than the other two algorithms

Table 2 The w_Up matrix, $n = 2,392$, $tol = 10^{-14}$

α	0.50	0.70	0.85	0.99
Jacobi	40 (0.11)	76 (0.23)	167 (0.52)	2695 (8.13)
PCG	23 (0.08)	31 (0.14)	43 (0.16)	69 (0.25)
GeneRank	(6.61)	(6.70)	(6.72)	(6.70)

Example 1: the number of matrix-vector products and the CPU time in seconds (in *brackets*) of the three algorithms. We see that PCG works better than the other two algorithms

the number of matrix-vector products and CPU time, even when α is medium, say $\alpha = 0.5$. Especially, when α is close to 1, the new algorithm works much better than the other two. For example, for the w_All matrix, when $\alpha = 0.99$, the GeneRank algorithm and the Jacobi iteration need 31.1 and 20.1 s, respectively, while the PCG method only uses 0.59 s, to reach the desired accuracy. Figure 1 plots convergence curves of the Jacobi iteration and the PCG method on the w_All matrix. One can see that PCG uses much fewer matrix-vector products than the Jacobi iteration, to reach the desired accuracy of 10^{-14}. It turns out that the new algorithm performs much better than the classical Jacobi iteration for GeneRank.

Example 2 In this example, we compare the PCG method with the Arnoldi algorithm and the modified Arnoldi algorithm proposed recently (Wu et al. 2010). The test matrix is the SNPa matrix, which is of size $152,520 \times 152,520$, with 639, 248 nonzero elements. The structure of the matrix is plotted in Fig. 2. In this experiment, we set $\mathbf{ex} = ones(n, 1)/n$, where $ones(n, 1)$ is an n-dimensional vector whose elements are all ones, and $n = 152,520$ is the size of the matrix in question.

In practice, for the sake of the growth of memory and the computational cost required, the Arnoldi and the modified Arnoldi algorithms will become impractical when m (the steps of the Arnoldi process) is large. Recall that we only need to store 6 vectors in the PCG method, while to store $m + 3$ vectors of size n in the Arnoldi algorithm; and the PCG method is cheaper than two Arnoldi-type algorithms. From now on, we set $m = 3$ in the Arnoldi and the modified Arnoldi algorithms, and pick $v_1 = \mathbf{ex}/||\mathbf{ex}||_2$ to be the initial vector for the Arnoldi process. The stopping criterion in the Arnoldi algorithm and the modified Arnoldi algorithm is

$$\|G\tilde{x} - \tilde{x}\|_1 /||\tilde{x}||_1 \leq tol, \tag{3.3}$$

with $tol = 10^{-10}$; and $\tilde{x}$ are some approximations to the GeneRank vector.

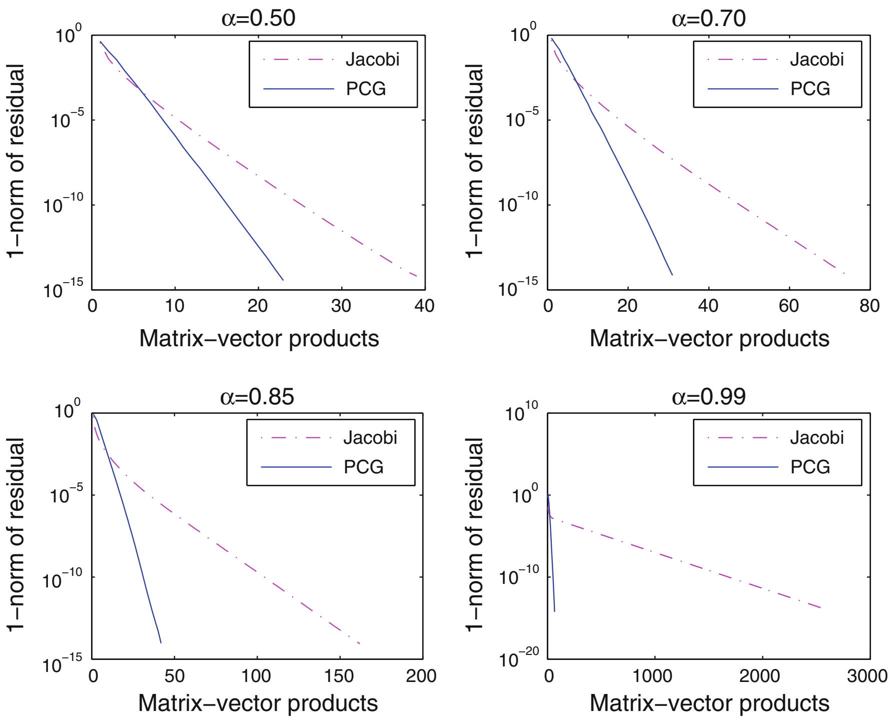
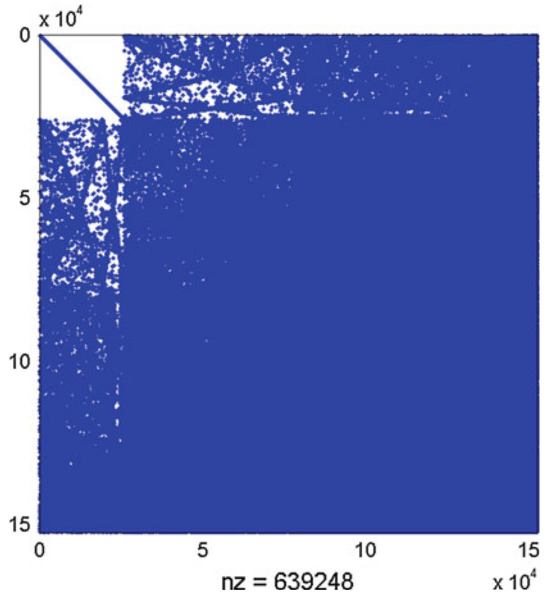


Fig. 1 Example 1, convergence curves of the Jacobi iteration and the PCG method on the *w_All* matrix. It is seen that PCG uses much fewer matrix-vector products than the Jacobi iteration, to reach the desired accuracy

Fig. 2 Example 2, structure of the SNPa matrix (plotted by the "spy" function in MATLAB), where the *stripe* represents nonzero elements, and *white* stands for zero elements



In this example, we also present the numerical results obtained from running the Jacobi iteration, where the initial guess for the Jacobi iteration and the PCG method is set to be zero, and the stopping criteria for the two algorithms are (3.1) and (3.2), respectively. Table 3 lists the numerical results of the five algorithms.

We mention that, for this test problem, the GeneRank algorithm does not work at all, due to the heavy storage requirements and the computational cost. It is shown that the three Krylov subspace algorithms use fewer matrix-vector products than the Jacobi iteration when α is close to 1, while the PCG method is superior to the other algorithms in terms of the number of matrix-vector multiplications and the CPU time. Specifically, in Fig. 3 we plot convergence curves of the four iterative algorithms when the damping factor $\alpha = 0.85$ (the damping factor used by Google, Page et al. 1998). It is obvious to see that the modified Arnoldi algorithm and the PCG method converge much faster than the other two algorithms, while the PCG algorithm works the best.

Table 3 The SNPa matrix, $n = 152,520$, $tol = 10^{-10}$

α	0.50	0.70	0.85	0.99
Jacobi	29 (1.89)	52 (3.39)	109 (7.08)	1752 (113.9)
Arnoldi	30 (3.22)	45 (4.80)	75 (8.23)	435 (47.8)
Modi-Arn	20 (2.00)	28 (2.83)	44 (4.45)	132 (13.9)
PCG	18 (1.81)	25 (2.56)	38 (3.86)	112 (11.4)
GeneRank	nw	nw	nw	nw

Example 2: the number of matrix-vector products and the CPU time in seconds (in *brackets*) of the five algorithms, $tol = 10^{-10}$. We observe that PCG performs better than the other four algorithms, where nw means "does not work".

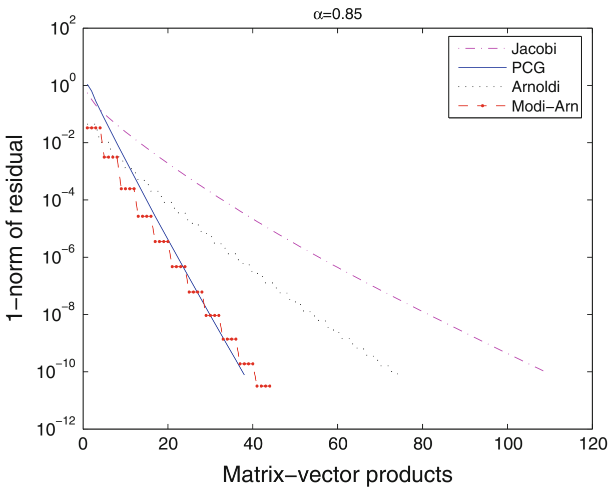


Fig. 3 Example 2, convergence curves of the four iterative algorithms on the SNPa matrix, $\alpha = 0.85$, $tol = 10^{-10}$. We see that the PCG and the modified Arnoldi algorithms use much fewer matrix-vector products than the other two algorithms, and the PCG algorithm works the best

Example 3 The aim of this example is three-fold. We first compare the PCG method with the CG method for GeneRank, and show superiority of the former. Then, we compare the actual output of the algorithms. More precisely, we show how correlated are the approximate GeneRank vectors from the different methods. Finally, we illustrate sharpness of Theorem 3. The test matrix is the w_Down matrix (see footnote 2), which is of size $1,625 \times 1,625$, with $67,252$ nonzero elements, and the experimental data is $\mathbf{ex} = extr\_dataDown$. Table 4 lists the numerical results. As a comparison, we also present the numerical results from running the GeneRank algorithm, the Jacobi iteration, the Arnoldi algorithm and the modified Arnoldi algorithm.

Again, we observe from the numerical results that the iterative algorithms are preferable, compared with the GeneRank algorithm; while the PCG method beats the other five algorithms. The Jacobi iteration converges faster than the CG method when α is far away from 1. However, when α is close to 1, the CG method outperforms the Jacobi iteration in many cases.

In order to see how correlated are the approximate GeneRank vectors from the different methods, we present the sine of the angles between the approximations obtained from the iterative algorithms and the "exact" solution obtained from the GeneRank algorithm. Here the sine of the angle between two vectors are evaluated as follows: Let x and y be two (nonzero) real vectors, then (Golub and Loan 1996)

$$\sin\angle(x,y) = \left\| \left(I - \frac{xx^T}{||x||_2\,||x^T||_2} \right) \frac{y}{||y||_2} \right\|_2 = \left\| \frac{y}{||y||_2} - \frac{x}{||x||_2} \left(\frac{x^T y}{||x^T||_2\,||y||_2} \right) \right\|_2 . \tag{3.4}$$

Recall that the smaller the sine of the angle, the more correlated the two vectors. Table 5 lists the numerical results, in which x^* denotes the "exact" solution from running the GeneRank algorithm, while $x^J,\ x^G,\ x^M,\ x^{CG}$ and x^{PCG} are the approximations from running the Jacobi iteration, the Arnoldi algorithm, the modified Arnoldi algorithm, the CG algorithm, as well as our PCG algorithm. We observe that the angles between the GeneRank vectors from the different methods approach the machine precision $\epsilon \approx 2.22 \times 10^{-16}$ (note that the stopping criterion is $tol = 10^{-14}$ for the iterative algorithms), implying that the final GeneRank vectors from different methods are

Table 4 The w_Down matrix, $n = 1,625$, $tol = 10^{-14}$

α	0.50	0.70	0.85	0.99
Jacobi	39 (0.06)	74 (0.13)	162 (0.27)	2610 (4.34)
Arnoldi	42 (0.11)	69 (0.17)	117 (0.31)	318 (0.84)
Modi-Arn	28 (0.08)	37 (0.09)	60 (0.14)	116 (0.30)
CG	228 (0.45)	261 (0.52)	299 (0.58)	411 (0.80)
PCG	23 (0.05)	32 (0.06)	45 (0.09)	73 (0.14)
GeneRank	(2.23)	(2.23)	(2.25)	(2.25)

Example 3: the number of matrix-vector products and the CPU time in seconds (in *brackets*) of the four algorithms, $tol = 10^{-14}$. We observe that PCG performs better than the other algorithms.

Table 5 A comparison of the actual output of the algorithms

α	0.50	0.70	0.85	0.99
$\sin\angle(x^J, x^*)$	1.9660×10^{-14}	4.6380×10^{-14}	5.4735×10^{-14}	6.2436×10^{-14}
$\sin\angle(x^G, x^*)$	1.1167×10^{-14}	1.5734×10^{-14}	3.9962×10^{-14}	2.8546×10^{-13}
$\sin\angle(x^M, x^*)$	3.1658×10^{-15}	2.0773×10^{-14}	2.0635×10^{-14}	5.0275×10^{-13}
$\sin\angle(x^{CG}, x^*)$	6.4563×10^{-15}	4.6642×10^{-15}	3.5567×10^{-15}	3.2223×10^{-15}
$\sin\angle(x^{PCG}, x^*)$	6.0975×10^{-15}	5.6169×10^{-15}	2.8789×10^{-15}	3.4990×10^{-15}

Example 3: correlation of the approximate GeneRank vectors from different methods. The w_Down matrix, $n = 1,625$. It is seen that the approximations are correlated

highly correlated. Moreover, it seems that the approximations obtained from the CG and PCG algorithms are more accurate than those from other three algorithms when α is large, say, 0.99.

So as to show effectiveness of Theorem 3, in Fig. 4 we depict (in decreasing order) the eigenvalues of $D - \alpha W$ and those of the preconditioned matrix $I - \alpha \cdot D^{-1/2} W D^{-1/2}$ when $\alpha = 0.5$. Here the x-label stands for the order of the eigenvalues, while the y-label represents the magnitude of the eigenvalues. We find numerically that $\lambda_{1,625}(D - \alpha W) = \lambda_{\min}(D - \alpha W) = 0.5$, $\lambda_1(D - \alpha W) = \lambda_{\max}(D - \alpha W) = 193.4$ (the right lower most and the left upper most "*" in the left figure, respectively); while $\lambda_{1,625}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = \lambda_{\min}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = 0.5$ and $\lambda_1(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = \lambda_{\max}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = 1.5$ (the right lower

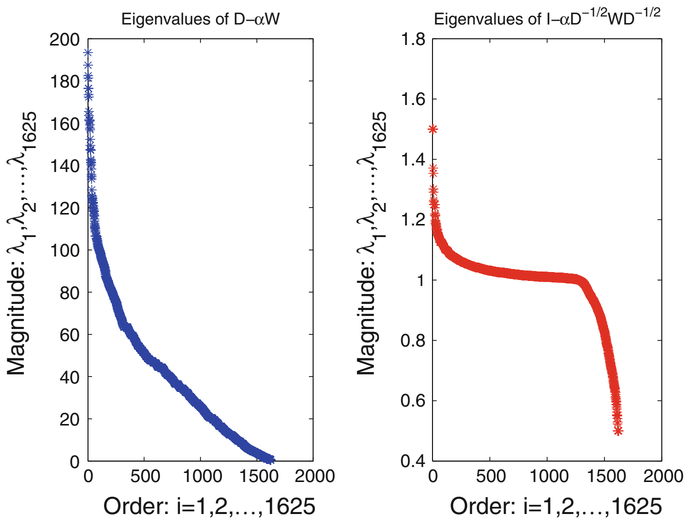


Fig. 4 Example 3, eigenvalues of $D - \alpha W$ and those of $I - \alpha \cdot D^{-1/2} W D^{-1/2}$ when $\alpha = 0.50$. We find numerically that $\lambda_{\min}(D - \alpha W) = 0.5$, $\lambda_{\max}(D - \alpha W) = 193.4$ (the *right lower most* and the *left upper most* "*" in the *left figure*, respectively); while $\lambda_{\min}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = 0.5$ and $\lambda_{\max}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = 1.5$ (the *right lower most* and the *left upper most* "*" in the *right figure*, respectively). This illustrates that Theorem 3 is very sharp

most and the left upper most "*" in the right figure, respectively). This illustrates that Theorem 3 is very sharp. Moreover, this example explains why the PCG method is superior to the CG method. It is seen from Table 4 that when $\alpha = 0.5$, the CG method uses 228 matrix-vector products while the PCG method only uses 23 matrix-vector products to derive the desired accuracy, and the improvement is impressive. Indeed, we have $\kappa_2(D - \alpha W) = 386.8$, while $\kappa_2(I - \alpha \cdot D^{-1/2} W D^{-1/2}) = 3$, so it follows from Theorem 2 and Corollary 1 that PCG can work much better than the CG method.

Example 4 Predicting protein function at the proteomic-scale is a key task in computational systems biology. High-throughout experimental methods have recently made available many protein interaction networks that need to be analyzed in order to provide insight into the functional role of proteins in the organization of the cell. There are two aims for this experiment. Firstly, we show that the PCG method can also be applied to protein function prediction in PPI networks. Secondly, we try to show efficiency of the PCG method for large scale biological computational problems.

Yeast two hybrid PPI networks have proteins as nodes. Two nodes share an undirected edge if they have been experimentally observed to interact (Xenarios et al. 2002). In[3], Grindrod proposed and analyzed a random graph model, now referred to as RENGA. In this model, nodes are considered to lie at unit intervals on a line, and they have a natural linear ordering, $i = 1, 2, \ldots, n$. Independently over all pairs of nodes, we then insert a link between nodes i and j with probability $\beta\lambda^{|j-i|-1}$, where $\beta > 0$ and $0 < \lambda < 1$ are fixed parameters. Taylor and Higham (2008) presented the MATLAB code of the RENGA model, whose MATLAB file is available from (see footnote 3). It returns an n by n symmetric nonnegative matrix from the function call $A = renga(n, \lambda, \beta)$.

Despite many approaches have been proposed in the recent years, the problem of protein function prediction in protein interaction networks is still open to substantial improvements and can be considered far from being solved (Sharan et al. 2007). Recently, Freschi (2007) proposed a modified application of the GeneRank algorithm, i.e., the ProteinRank algorithm, to the problem of protein function prediction in PPI networks. The key of the ProteinRank algorithm Freschi (2007) resorts to solving a series of linear systems which are of the same type as (1.4). So as to show our new algorithm is preferable to large problems, we run the *renga* code with $n = 10^5,\ 5 \times 10^5$, and 10^6, respectively, and set $\mathbf{ex} = ones(n, 1)/n$; with λ defaulting to 0.9 and β defaulting to 1 (Taylor and Higham 2008). Note that the choice $\beta = 1$ ensures that adjacently ordered nodes are always connected. We run the GeneRank algorithm as well as the five iterative algorithms on these large problems. Tables 6, 7, 8 list the numerical results.

For this example, the GeneRank algorithm does not work at all, due to the fact that the sizes of the matrices are very large. Furthermore, in order to evaluate efficiency of the PCG method with respect to the Jacobi iteration, we define

$$\text{Efficency} = \frac{\text{CPU}_{\text{Jacobi}} - \text{CPU}_{\text{PCG}}}{\text{CPU}_{\text{Jacobi}}}, \tag{3.5}$$

[3] http://www.maths.strath.ac.uk/research/groups/numerical_analysis/contest. Accessed 2010.

Table 6 $n = 100,000$, $tol = 10^{-8}$

α	0.50	0.70	0.85	0.99
Jacobi	16 (0.99)	28 (1.74)	56 (3.49)	592 (36.8)
Arnoldi	15 (1.53)	24 (2.45)	42 (4.33)	273 (27.9)
Modi-Arn	12 (1.16)	20 (1.94)	32 (3.06)	96 (9.48)
CG	18 (1.44)	21 (1.67)	29 (2.31)	106 (8.44)
PCG	11 (0.94)	15 (1.28)	23 (1.95)	93 (7.92)
GeneRank	nw	nw	nw	nw

Table 7 $n = 500,000$, $tol = 10^{-8}$

α	0.50	0.70	0.85	0.99
Jacobi	16 (5.14)	28 (9.00)	55 (17.7)	591 (190.0)
Arnoldi	15 (7.86)	24 (12.8)	42 (22.3)	264 (139.3)
Modi-Arn	12 (6.03)	20 (10.1)	32 (15.8)	100 (51.1)
CG	17 (6.98)	21 (8.66)	29 (11.9)	106 (43.9)
PCG	11 (4.84)	15 (6.61)	23 (10.2)	92 (40.7)
GeneRank	nw	nw	nw	nw

Table 8 $n = 1,000,000$, $tol = 10^{-8}$

α	0.50	0.70	0.85	0.99
Jacobi	16 (9.61)	28 (16.9)	56 (33.7)	603 (364.4)
Arnoldi	18 (17.7)	24 (23.5)	42 (41.9)	258 (257.3)
Modi-Arn	12 (11.8)	20 (20.0)	32 (38.1)	112 (114.1)
CG	20 (16.5)	22 (18.2)	30 (24.8)	110 (91.0)
PCG	11 (9.36)	16 (13.6)	24 (20.3)	94 (79.7)
GeneRank	nw	nw	nw	nw

Example 4: the number of matrix-vector products and the CPU time in seconds (in brackets) of the six algorithms. It is seen that PCG outperforms the other algorithms, especially when the size of the matrix is large. Here nw means "does not work"

whose values are listed in Table 9. We observe that, for the same matrix, the closer the damping factor is to 1, the larger the values of "Efficiency". On the other hand, for a given damping factor, the values of "Efficiency" are comparable. In other words, the efficiency of the PCG method is consistent even though the size of the problem increases. Therefore, our new algorithm is favorable to large scale biology computational problems, in which the computational cost is often very high.

4 Conclusions and future work

With the rapid growth in biological data sources containing gene-related information and microarray expression data, there has been much interest in recent years

Table 9 Efficiency of the PCG algorithm

Example 4: efficiency of PCG relative to the Jacobi iteration by (3.5)

α	0.50	0.70	0.85	0.99
$n = 10^5$	5.05%	26.4%	44.1%	78.5%
$n = 5 \times 10^5$	5.84%	26.6%	42.4%	78.6%
$n = 10^6$	2.60%	19.5%	39.8%	78.1%

in developing bioinformatics approaches. These approaches can be used to analyze and to identify important genes. GeneRank is an intuitive modification of PageRank that maintains many of its mathematical properties. The GeneRank model combines gene expression information with a network structure derived from gene ontologies or expression profile correlations. In Morrison et al. (2005), the computation of GeneRank resorts to solving a large unsymmetric matrix computation problem. However, when the size of the matrix involved is very large, the GeneRank algorithm may be ineffective and even may be infeasible. Moreover, the adjacency matrix is symmetric in the GeneRank model, while the original GeneRank algorithm fails to exploit this property.

In this paper, we point out that the GeneRank problem can be rewritten as a SPD linear system, so that the structure of the adjacent matrix can be used efficiently. Some mathematical properties of the new coefficient matrix are analyzed. Then, we propose a preconditioned CG method to solve the new system; and we theoretically prove that the new algorithm is superior to the unpreconditioned CG method. Finally, numerical experiments support our theoretical results, and show superiority of our new algorithm over the other algorithms for computing GeneRank. However, there is still much work needs to be performed. For instance, how to determine the *optimal* damping factor α? Can we propose more efficient mathematical models for ranking genes? They are interesting topics and deserve further investigation.

Acknowledgements We would like to express our sincere thanks to the reviewers for their invaluable suggestions that make us greatly improve the representation of this paper. Meanwhile, we are grateful to J. Morrison, R. Breitling, D. Higham, D. Gilbert and Jun-feng Yin, for providing us with the GeneRank data files utilized in the numerical experiments. We also appreciate A. Taylor and D. Higham for providing us with the MATLAB file of the RENGA model. Gang Wu is supported by the National Science Foundation of China under grants 10901132 and 11171289, the Qing-Lan Project of Jiangsu Province, and the 333 Project of Jiangsu Province. The work of Wei Xu is partially supported by the National Science Foundation of China under grants 10901132 and 11101310. The work of Ying Zhang is partially supported by the National Science Foundation of China under grant 10901132. Yimin Wei is supported by the National Natural Science Foundation of China under grant 10871051, 973 Program Project (no. 2010CB327900), Doctoral Program of the Ministry of Education under grant 20090071110003, Shanghai Education Committee (Dawn Project 08SG01) and Shanghai Science and Technology Committee under grant 09DZ2272900.

5 Appendix

In this section, we review some iterative algorithms for computing GeneRank, and provide proofs of some theoretical results presented in Sect. 3. The MATLAB code of the PCG algorithm, and the codes of the modified Arnoldi algorithm for GeneRank (Wu et al. 2010) are also provided.

5.1 Some iterative algorithms for computing GeneRank

The GeneRank algorithm (Morrison et al. 2005) exploits a direct method for solving the linear system (1.4). However, when the matrix is very large and sparse, the direct method is inefficient, or it even may fail to work (Demmel 1997; Golub and Loan 1996). In practice, one can solve (1.4) using iterative algorithms.

One option is the Jacobi iteration (Demmel 1997; Golub and Loan 1996): Given a nonnegative vector $x^{(0)}$, we define the vector series

$$x^{(k)} = \alpha \cdot WD^{-1}x^{(k-1)} + (1-\alpha)\mathbf{ex}, \quad k = 1, 2, \ldots \tag{5.1}$$

Indeed, the iteration matrix of (5.1) is αWD^{-1}, whose spectral radius satisfies

$$\rho(\alpha WD^{-1}) \leq ||\alpha WD^{-1}||_1 \leq \alpha < 1, \tag{5.2}$$

thus the Jacobi iteration converges unconditionally, i.e., $x^{(k)} \rightarrow x^*$ as $k \rightarrow \infty$. Moreover, the smaller the damping factor is, the faster the Jacobi iteration converges (Demmel 1997). The framework of this algorithm is listed as follows.

Algorithm 4 The Jacobi iteration for GeneRank (Jacobi)

```
1. Start: Given the matrix $W$, the damping factor $\alpha$, and two vectors $\mathbf{ex}$ and $\mathbf{d} = [d_1, d_2, \ldots, d_n]^T$. Choose a unit positive starting vector $x^{(0)}$, and a prescribed tolerance tol, set $k = 1, r = 1$;
2. Iterate:
while $r > tol$
  % Here "./" denotes the right array divide commend
  $x^{(k)} = \alpha W(x^{(k-1)}./\mathbf{d}) + (1-\alpha)\mathbf{ex}$;
  $r = \|x^{(k)} - x^{(k-1)}\|_1$;
  $k = k + 1$;
end
```

In the above algorithm, we need to store four vectors $x^{(k)}$, $x^{(k-1)}$, $\mathbf{d}$ and $\mathbf{ex}$ in the main memory, and to perform one matrix-vector product per iteration, in $\mathcal{O}(n)$ flops. As the GeneRank vector is a stationary distribution of a Markov chain, we use the one-norm of residual as the stopping criterion throughout this paper. Unfortunately, it is well known that the Jacobi iteration may converge very slowly when the spectral radius of the iterative matrix is close to 1 (Demmel 1997; Golub and Loan 1996). In other words, when α is close to 1, Algorithm 4 may perform poorly, and it is necessary to seek new iterative algorithms for solving (1.4), especially when the matrix is very large.

The Krylov subspace method is one of the most popular methods for large scale matrix computations (Bai et al. 2000; Saad 2003). Recently, two Arnoldi-type methods for solving GeneRank were proposed in Wu et al. (2010). In these methods, the GeneRank problem is rewritten as an (unsymmetric) eigenvalue problem as follows (Wu et al. 2010; Yue et al. 2007):

$$\left[\alpha \cdot WD^{-1} + (1-\alpha)(\mathbf{ex} \cdot \mathbf{e}^T)\right] x^* = x^*, \tag{5.3}$$

where $\mathbf{e}$ is the vector of all ones. Then projection methods (Bai et al. 2000; Saad 2003) are used for computing the dominant eigenvector corresponding to the eigenvalue 1 of the GeneRank matrix

$$G \equiv \alpha \cdot WD^{-1} + (1-\alpha)(\mathbf{ex} \cdot \mathbf{e}^T), \tag{5.4}$$

which is nothing but the GeneRank vector.

More precisely, given an initial vector v_1 in a unit norm, if computations are performed in exact arithmetic, then the m-step Arnoldi process generates successively an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ for the Krylov subspace $\mathcal{K}_m(A, v_1) = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}$. In this subspace, the restriction of A is represented by an $m \times m$ upper Hessenberg matrix H_m with the entries $h_{i,j}$. The Arnoldi process is given as follows, for more details, refer to Bai et al. (2000) and Saad (2003).

Algorithm 5 The m-step Arnoldi process
1. *Start: Given the initial vector v_1 of unit norm, and the steps m of the Arnoldi process;*
2. *Iterate:*

```
for $j = 1, 2, \ldots, m$
  $q = Av_j$;
  for $i = 1, 2, \ldots, j$
    $h_{i,j} = v_i^T q$;
    $q = q - h_{i,j} v_i$;
  end for
  $h_{j+1,j} = ||q||_2$;
  if $h_{j+1,j} = 0$
    break;
  end if
  $v_{j+1} = q / h_{j+1,j}$;
end for
```

Furthermore, we have (Bai et al. 2000; Saad 2003)

$$GV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T = V_{m+1} \tilde{H}_m, \tag{5.5}$$

where e_m is the m-th coordinate vector of dimension m; and $\tilde{H}_m$ is an $(m+1) \times m$ upper Hessenberg matrix which is the same as H_m except for an additional row whose unique nonzero entry is $h_{m+1,m}$.

The Arnoldi algorithm for the GeneRank problem is inspired by the refined strategy (Jia 1997) and the fact that the largest eigenvalue of the GeneRank matrix is 1. In this algorithm, a unit norm vector $x^G \in \mathcal{K}_m(G, v_1)$ is found to satisfy

$$\left\|(G-I)x^G\right\|_2 = \min_{\substack{u \in \mathcal{K}_m(G, v_1), \\ ||u||_2 = 1}} ||(G-I)u||_2. \tag{5.6}$$

A restarted version of the Arnoldi algorithm for GeneRank is presented as follows.

Algorithm 6 Wu et al. (2010) An Arnoldi algorithm for computing GeneRank (Arnoldi)
1. *Start: Given an initial guess v_1 of unit norm, the Arnoldi steps number m, as well as a prescribed tolerance tol; Set $\rho = 1$;*
2. *Iterate*

```
while $\rho > tol$
    Run Algorithm 5 for the computation of $V_{m+1}$ and $\tilde{H}_m$;
    Compute singular value decomposition (SVD):
    $\tilde{H}_m - [I; \mathbf{O}] = U \Gamma S^T$;
    Approximate eigenvector: $v_1 = V_m S(:, m)$;
    Compute the residual norm $\rho$ with respect to the
    approximate eigenvector:
     $\rho = \frac{||V_{m+1}[\tilde{H}_m S(:,m)] - V_m S(:,m)||_1}{||V_m S(:,m)||_1}$;
end
```

Note that the m-step Arnoldi process builds an orthonormal basis $V_{m+1} = [v_1, v_2, \ldots, v_m, v_{m+1}]$ for Krylov subspace $\mathcal{K}_{m+1}(G, v_1)$. However, the approximation obtained from the Arnoldi algorithm is in $\mathcal{K}_m(G, v_1) = span\{v_1, v_2, \ldots, v_m\}$. Therefore, a modified Arnoldi algorithm is presented to accelerate the convergence of Algorithm 6, using the $(m+1)$-th basis vector v_{m+1} (Wu et al. 2010). That is, one seeks a vector $x^G + \hat{\beta} v_{m+1} \in \mathcal{K}_{m+1}(G, v_1)$ that satisfies the following optimal property

$$\left\|(G - I)(x^G + \hat{\beta} v_{m+1})\right\|_2 = \min_{\beta \in \mathbb{C}} \left\|(G - I)\left(x^G + \beta v_{m+1}\right)\right\|_2. \tag{5.7}$$

This algorithm is outlined as follows, for more detail, refer to Wu et al. (2010).

Algorithm 7 Wu et al. (2010) A modified Arnoldi algorithm for computing GeneRank (Modi-Arn)
1. *Start: Given an initial guess v_1 of unit norm, the Arnoldi steps number m, as well as a prescribed tolerance tol; Set $\rho = 1$;*
2. *Iterate*

```
while $\rho > tol$
    Run Algorithm 3 for the computation of $V_{m+1} = [V_m, v_{m+1}]$ and $\tilde{H}_m$;
    Compute SVD: $\tilde{H}_m - [I; \mathbf{O}] = U \Gamma S^T$, and set $\rho = \sigma_{\min}(\tilde{H}_m - [I; \mathbf{O}])$;
    Compute the new approximation $x^M$:
        $w = G v_{m+1} - v_{m+1}$, $\gamma = ||w||_2^2$;
        $\hat{\beta} = -\rho \cdot w^T \left[V_{m+1} \cdot U(:, m)\right] / \gamma$;
        % $x^M$, the new approximation
        $v_1 = V_{m+1} \cdot \left([S(:, m); \hat{\beta}]\right)$;
        $w = w + v_{m+1}$;
        $w = V_{m+1}[\tilde{H}_m S(:, m)] + \hat{\beta} w$;
        % $\rho = \left\|G x^M - x^M\right\|_1 / \left\|x^M\right\|_1$,
        $\rho = ||w - v_1||_1 / ||v_1||_1$;
        % $Gx^M / \|Gx^M\|_2$, the initial vector for the next Arnoldi iteration
        $v_1 = w/||w||_2$;
end
```

In addition to two vectors **ex** and **d**, Algorithm 6 needs to store $m+1$ vectors for the basis, and to perform m matrix-vector products per iteration; while Algorithm 7 needs to perform $m+1$ matrix-vector products per iteration, and to store $m+2$ vectors. Therefore, both Algorithms 6 and 7 have higher computational complicities than the Jacobi iteration. In practical calculations, we suggest choosing a medium m for the Arnoldi-type algorithms, say, $m = 3$ or 5.

5.2 Proof of Theorem 1

Proof It is obvious to see that $D - \alpha W$ is symmetric. To show the definiteness, it is sufficient to show that

$$D^{-1/2}(D - \alpha W)D^{-1/2} = I - \alpha(D^{-1/2}WD^{-1/2}) \tag{5.8}$$

is SPD. Note that $D^{-1/2}WD^{-1/2}$ and WD^{-1} share the same eigenvalues, we have

$$\rho(D^{-1/2}WD^{-1/2}) = \rho(WD^{-1}) \le ||WD^{-1}||_1 = 1, \tag{5.9}$$

where $\rho(WD^{-1})$ and $||WD^{-1}||_1$ denotes the spectral radius and one-norm of WD^{-1} (Demmel 1997; Golub and Loan 1996), respectively. As a result, all the eigenvalues of $I - \alpha(D^{-1/2}WD^{-1/2})$ are positive, so $D^{-1/2}(D - dW)D^{-1/2}$ is SPD. □

5.3 Proof of Theorem 3

Proof Let μ be an eigenvalue of $D^{-1/2}WD^{-1/2}$, then $\lambda = 1 - \alpha\mu$ is an eigenvalue of $I - \alpha \cdot D^{-1/2}WD^{-1/2}$. Moreover, the eigenvalues of $D^{-1/2}WD^{-1/2}$ are the same as those of WD^{-1}. So it is sufficient to check the largest and the smallest eigenvalue of WD^{-1}, respectively.

Denote by $\rho(WD^{-1})$ the spectral radius of WD^{-1}, then

$$\rho(WD^{-1}) \le ||WD^{-1}||_1 = 1. \tag{5.10}$$

Therefore,

$$\lambda_{\max}(WD^{-1}) \le 1, \quad \lambda_{\min}(WD^{-1}) \ge -1, \tag{5.11}$$

that is,

$$\lambda_{\max}(D^{-1/2}WD^{-1/2}) \le 1, \tag{5.12}$$
$$\lambda_{\min}(D^{-1/2}WD^{-1/2}) \ge -1. \tag{5.13}$$

Thus, it leads to

$$\lambda_{\max}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) \leq 1 + \alpha, \tag{5.14}$$

and

$$\lambda_{\min}(I - \alpha \cdot D^{-1/2} W D^{-1/2}) \geq 1 - \alpha. \tag{5.15}$$

This completes the proof. □

5.4 Proof of Corollary 1

Proof Recall from Theorem 1 that $I - \alpha \cdot D^{-1/2} W D^{-1/2}$ is SPD. So we have from the convergence theory of PCG method that (Demmel 1997; Golub and Loan 1996; Saad 2003)

$$||\hat{x}^* - x^{(k)}||_{\widehat{D}} \leq 2\left(\frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1}\right)^k ||\hat{x}^* - x^{(0)}||_{\widehat{D}}, \tag{5.16}$$

where

$$\begin{aligned} \kappa_2 &\equiv \kappa_2\big(I - \alpha \cdot D^{-1/2} W D^{-1/2}\big) \\ &= ||I - \alpha \cdot D^{-1/2} W D^{-1/2}||_2 \cdot \|\big(I - \alpha \cdot D^{-1/2} W D^{-1/2}\big)^{-1}||_2 \end{aligned} \tag{5.17}$$

is the two-norm condition number of the preconditioned matrix $I - \alpha \cdot D^{-1/2} W D^{-1/2}$. So we have from Theorem 3 that

$$\kappa_2\big(I - \alpha \cdot D^{-1/2} W D^{-1/2}\big) = \frac{\lambda_{\max}(I - \alpha \cdot D^{-1/2} W D^{-1/2})}{\lambda_{\min}(I - \alpha \cdot D^{-1/2} W D^{-1/2})} \leq \frac{1+\alpha}{1-\alpha}. \tag{5.18}$$

Thus, (2.9) follows from (5.16), and the fact that $(\sqrt{\kappa_2} - 1)/(\sqrt{\kappa_2} + 1)$ is a monotonously increasing function on κ_2. □

5.5 Proof of Theorem 4

Proof (i) It is known that (Golub and Loan 1996)

$$\sin \angle(x^{(k)}, x^*) = \min_{\beta \in \mathbb{C}} ||x^{(k)} - \beta x^*||_2 \leq ||x^{(k)} - x^*||_2. \tag{5.19}$$

We obtain from (5.1) that

$$\begin{aligned} ||x^{(k)} - x^*||_2 &\leq ||(\alpha W D^{-1})^k (x^{(0)} - x^*)||_2 \\ &\leq ||(\alpha W D^{-1})^k||_2 ||x^{(0)} - x^*||_2 \\ &\leq \sqrt{n} ||(\alpha W D^{-1})^k||_1 ||x^{(0)} - x^*||_2 \\ &= \alpha^k \cdot \sqrt{n} ||x^{(0)} - x^*||_2, \end{aligned}$$

where we use the relations that $||B||_2 \leq \sqrt{n}||B||_1$ for any matrix B of size n (Golub and Loan 1996), and $||(WD^{-1})^k||_1 = 1,\ k = 1, 2, \ldots$. As a result, $\sin\angle(x^{(k)}, x^*) \to 0$ as $k \to \infty$.

(ii) The result is from Theorem 2.2 of Wu et al. It shows that when α is not too close to 1, $\sin\angle(x^G, x^*)$ and $\sin\angle(x^M, x^*)$ will tend to zero as soon as the distance between x^* and the search subspaces approaches zero.

(iii) We only need to prove (2.16), the proof of (2.15) is similar. Indeed,

$$\begin{aligned}||\hat{x}^* - x^{(k)}||_{\widehat{D}}^2 &= (\hat{x}^* - x^{(k)})^T\widehat{D}(\hat{x}^* - x^{(k)})\\ &\geq \sigma_{\min}(\widehat{D}) \cdot ||\hat{x}^* - x^{(k)}||_2^2\\ &\geq \sigma_{\min}(\widehat{D}) \cdot \sin^2\angle(\hat{x}^*, x^{(k)})\\ &= \lambda_{\min}(\widehat{D}) \cdot \sin^2\angle(\hat{x}^*, x^{(k)}),\end{aligned}$$

where $\sigma_{\min}(\widehat{D})$ stands for the smallest singular value of $\widehat{D}$. Consequently, it follows from Theorem 3 that

$$\sin\angle(\hat{x}^*, x^{(k)}) \leq \frac{1}{\sqrt{\lambda_{\min}(\widehat{D})}}||\hat{x}^* - x^{(k)}||_{\widehat{D}} \leq \frac{1}{\sqrt{1-\alpha}}||\hat{x}^* - x^{(k)}||_{\widehat{D}}.$$

Again, it shows that $\sin\angle(\hat{x}^*, x^{(k)}) \to 0$ as $||\hat{x}^* - x^{(k)}||_{\widehat{D}} \to 0$, when α is not very close to 1.

□

5.6 MATLAB code of the PCG algorithm for GeneRank

```
function [x,resvec] = pcgGR(W,ex,d,tol,maxit);
%++++++++++++++++++++++++++++++++++++++++++++++++++++
% PCG for GeneRank
% INPUT:
% W is the GeneRank matrix
% ex is the righthand side
% d is the damping factor
% tol is the convergence tolerance
% maxit is the the maximal number of iterations
% OUTPUT:
% x is the GeneRank vector
% resvec is a vector whose elements are the 1-residual norms during iterations
% For example, [x,resvec] = pcgGR(W,ex,0.85,1e − 8, 1e + 5)
%++++++++++++++++++++++++++++++++++++++++++++++++++++
% Set up for the method
ex = abs(ex);
ex=ex/sum(ex);
deg = sum(W);
ind = find(deg == 0);
```

```
deg(ind) = 1;
n=size(W,1);
deg=deg';    % D = sparse(diag(deg));
x = zeros(n,1);    % the initial guess
r=ex;    % zero-th residual
normr=sum(abs(r));    % norm of 1-residual
resvec = [ ];    % storing residual norm
mv=0;    % the number of matrix-vector products
rho = 1;
stag = 0;    % stagnation of the method
t = cputime;
% loop over maxit iterations (unless convergence or failure)
for i = 1 : maxit
  z = r./deg;
  rho1 = rho;
  rho = r' * z;
  if ((rho == 0)| isinf(rho))
    break;
  end
  if (i == 1)
    p = z;
  else
  beta = rho / rho1;
  if ((beta == 0) | isinf(beta))
    break;
  end
  p = z + beta*p;
  end
  q =deg. * p - d * (W * p);    % the matrix-vector product
  mv=mv+1;    % number of matrix-vector products
  pq = p' * q;
  if ((pq <= 0) | isinf(pq))
    break;
  else
    alpha = rho / pq;
  end
  if  isinf(alpha)
    break;
  end
  if (alpha == 0)    % stagnation of the method
    stag = 1;
  end
  % check for stagnation of the method
  if (stag == 0)
    stagtest = zeros(n,1);
    ind = (x ~ = 0);
    stagtest(ind) = p(ind)./ x(ind);
    stagtest( ind & p  = 0) = Inf;
    if (abs(alpha) * norm(stagtest, inf) < eps)
      stag = 1;
    end
  end
  x = x + alpha * p;      % new approximation
  r = r - alpha * q;
  normr=sum(abs(r));      % 1-residual norm
  resvec=[resvec;normr];
  if (normr <= tol)      % check for convergence
    flag = 0;
    iter = i;
    break;
  end
  if (stag == 1)
    break;
  end
end      % for i = 1 : maxit
t = cputime - t      % CPU time of the algorithm
x = deg. * x;      % the computed solution
% plot the convergence curve
semilogy(resvec)
title('PCG for GeneRank')
xlabel('Number of matrix-vector products')
ylabel('1-residual norm')
```

5.7 MATLAB codes of the modified Arnoldi algorithm for GeneRank

```
function [v,res]=modiArnGR(W,ex,d,m,tol,maxmv);
%++++++++++++++++++++++++++++++++++++++++++++++++++++
% The modified Arnoldi algorithm for GeneRank
% G. Wu, Y. Zhang, and Y. Wei, Krylov subspace algorithms for computing
GeneRank for the analysis % of microarray data mining, J. Comput. Biology, 17:
631-646, 2010.
%INPUT:
% W is the GeneRank matrix
% ex is the righthand side
% d is the damping factor
% m is the steps of the Arnoldi process
% tol is the convergence tolerance
% maxmv is the maximal number of matrix-vector products
%OUTPUT:
% v is the computed GeneRank vector
% res is a vector whose elements are the 1-residual norms during iterations
% For example, [v,res]=modiArnGR(W,ex,0.85,3,1e-8,1e + 5);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++
% Set up for the method
ex = abs(ex);
ex=ex/sum(ex);
deg = sum(W);
ind = find(deg == 0);
deg(ind) = 1;
deg=deg';
w=ex;   % the initial vector
res=[ ];   % for storing residual norms
mv=0;   % the number of matrix-vector products
r=1;   % the initial residual norm
t1=clock;   % CPU time
% loop over maxit iterations (unless convergence or failure)
while r>tol & mv<=maxmv
  mv1=mv;   % number of matrix-vectors of the previous iterations
  [v,H,m,mv]=GArnproc(W,deg,ex,w,d,m,mv);   %run the Arnoldi process
  if H(m+1,m)<1e-14   % lucky breakdown
    [evr,ev]=eig(H(1:m,1:m));
    [ev,ind]=max(diag(ev));
    v = v(:, 1 : m) * evr(:, ind);   % computed solution
    return;
  end
  [u,s,t]=svd(H-[eye(m);zeros(1,m)],0);
  r=min(diag(s));   % 2-residual norm
  %*********************************Compute the modified vector
  w=MvGR(W,deg,ex,v(:,m+1),d)-v(:,m+1);   % G * v_{m+1} - v_{m+1}
  mv=mv+1;   % the procedure requires one more mat-vec product
  beta = w' * (v * u(:, m));
  gamma=norm(w)^ 2;
  beta = -r * beta/gamma;
  w=w+v(:,m+1);
  w = v * (H * t(:, m)) + beta * w;
  v = v * ([t(:, m); beta]);   % the modified vector
  %*********************************Compute the modified vector
  r=sum(abs(w-v))/sum(abs(v));   % the 1-residual norm
  res = [resr * ones(1, mv - mv1)];
end %while
t1=etime(clock,t1)   % CPU time
v=v/sum(abs(v));
mv   %the total matrix-vector products
% plot the convergence curve
res=res';
semilogy(res,'-')
title('A modified Arnoldi-type algorithm for GeneRank');
```

```
xlabel('Matrix-vector multiplications');
ylabel('1-Residual norms');

function [v,H,m,mv]=GArnproc(W,deg,ex,v,d,m,mv);
% the Arnoldi process for the GeneRank problem
w=MvGR(W,deg,ex,v,d);     % matrix-vector product for GeneRank
mv=mv+1;
normw=norm(w);
H(1,1) = v' * w;
w = w - H(1,1) * v;
h=norm(w);
if h < 0.618 * normw     % reorthogonalization
  h = v' * w;
  w = w - h * v;
  H(1,1)=H(1,1)+h;
  h=norm(w);
end
if h<1e-14
  m=1;
  H(2,1)=h;
  return;
else
  H(2,1)=h;
  w=w/h;
  v=[v w];
end
for j=2:m
  w=MvGR(W,deg,ex,v(:,j),d);     % matrix-vector product
  mv=mv+1;
  normw=norm(w);
  H=[H;zeros(1,j)];
  if j<m
    H=[H zeros(j+1,1)];
  end
  for i=1:j
    H(i,j) = v(:,i)' * w;
    w = w - H(i,j) * v(:,i);
  end
  h=norm(w);
  if h < 0.618 * normw     % reorthogonalization
    for i=1:j
      h = v(:,i)' * w;
      w = w - h * v(:,i);
      H(i,j)=h+H(i,j);
    end
    h=norm(w);
  end
  if h<1e-14
    m=j;
    H(j+1,j)=h;
    return;
  else
    H(j+1,j)=h;
    w=w/h;
    v=[v w];
  end
end
clear w;
clear h;
```

function v=MvGR(W,deg,ex,v,d);
% the function for matrix-vector product in the Arnoldi process
$v = ((1 - d) * sum(v)) * ex + W * (d * (v./deg));$

References

Aerts S et al (2006) Gene prioritization through genomic data fusion. Nat Biotechnol 24:537–544

Agarwal S, Sengupta S (2009) Ranking genes by relevance to a disease. Proc LSS Comput Syst Bioinform Conf 8:37–46

Bai Z, Demmel J, Dongarra J, Ruhe A, van der Vorst H (2000) Templates for the solution of algebraic eigenvalue problems: a practical guide. SIAM, Philadelphia

Benzi M (2002) Preconditioning techniques for large linear systems: a survey. J Comput Phys 182:418–477

Cipra B (2000) The best of the 20th Century: editors name top 10 algorithms. SIAM News 33(4)

Demmel JW (1997) Applied numerical linear algebra. SIAM, Philadelphia

Franke L et al (2006) Reconstruction of a functional human gene network, with an application for prioritizing positional candidate genes. Am J Hum Genet 78:1011–1025

Freschi V (2007) Protein function prediction from interaction networks using a random walk ranking algorithm. IEEE international conference on bioinformatics and bioengineering, pp 42–48

Golub GH, Van Loan CF (1996) Matrix computations, 3rd edn. The Johns Hopkins University Press, Baltimore, London

Hai D, Lee W, Thuy H (2008) A PageRanking based method for identifying chracteristic genes of a disease. IEEE Int Conf Netw Sens Control 6–8:1496–1499

Hestenes M, Stiefel E (1952) Methods of conjugate gradients for linear systems. J Res Natl Bur Stand 49:409–436

Jia Z (1997) Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. Linear Algebra Appl 259:1–23

Ma X, Lee H, Wang L, Sun F (2007) CGI: a new approach for prioritizing genes by combining gene expression and protein–protein interaction data. Bioinformatics 23(2):215–221

Morrison J, Breitling R, Higham D, Gilbert D (2005) GeneRank: using search engine for the analysis of microarray experiments. BMC Bioinform 6:233–246

Page L, Brin S, Motwami R, Winograd T (1998) The PageRank citation ranking: bring order to the web, Technical report. Computer Science Department, Stanford University, Palo Alto

Saad Y (2003) Iterative methods for sparse linear systems, 2nd edn. SIAM, Philadelphia

Sharan R, Ulitsky I, Shamir R (2007) Network-based prediction of protein function. Mol Syst Biol 3:1–13

Taylor A, Higham D (2008) CONTEST: a controllable test matrix toolbox for MATLAB. ACM Trans Math Softw 35, Article 26

The MATHWORKS, Inc. (2004) MATLAB 7. September

Wu G, Zhang Y, Wei Y (2010) Krylov subspace algorithms for computing GeneRank for the analysis of microarray data mining. J Comput Biol 17:631–646

Wu G, Zhang Y, Wei Y (under review) Accelerating the Arnoldi-type algorithm for computing Google's PageRank

Xenarios I, Salwinski L, Duan X, Higney P, Kim S (2002) DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. Nucleic Acids Res 30:303–305

Yue B, Liang H, Bai F (2007) Understanding the GeneRank model. IEEE 1st Int Conf Bioinform Biomed Eng 6–8:248–251