A Highly Efficient Parallel Algorithm for Computing the Fiedler Vector

Murat Manguoglu
Purdue University
March 22, 2010

Abstract
The eigenvector corresponding to the second smallest eigenvalue of the Laplacian of a graph, known as the Fiedler vector, has a number of applications in areas that include matrix reordering, graph partitioning, protein analysis, data mining, machine learning, and web search. The computation of the Fiedler vector has been regarded as an expensive process as it involves solving a large eigenvalue problem. We present a novel and efficient parallel algorithm for computing the Fiedler vector of large graphs based on the Trace Minimization algorithm (Sameh, et.al). We compare the parallel performance of our method with a multilevel scheme, designed specifically for computing the Fiedler vector, which is implemented in routine MC73Fiedler of the Harwell Subroutine Library (HSL). In addition, we compare the quality of the Fiedler vector for the application of weighted matrix reordering and provide a metric for measuring the quality of reordering.

1 Introduction
The second smallest eigenvalue and the corresponding eigenvector of the Laplacian of a graph have been used in a number of application areas including matrix reordering [11] [10] [9] [1], graph partitioning [14] [15], machine learning [13], protein analysis and data mining [5] [18] [8], and web search [4]. The second smallest eigenvalue of the Laplacian of a graph is sometimes called the algebraic connectivity of the graph, and the corresponding eigenvector is known as the Fiedler vector, due to the pioneering work of Fiedler [3].

For a given $n \times n$ sparse symmetric matrix $A$, or an undirected weighted graph with positive weights, one can form the weighted-Laplacian matrix, $L_w$, as follows:

$$L_w(i, j) = \left\{ \begin{array}{ll} \sum_j |A(i, j)| & \text{if } i = j, \\
-|A(i, j)| & \text{if } i \neq j. \end{array} \right.$$ (1)

One can obtain the unweighted Laplacian by simply replacing each nonzero element of the matrix $A$ by 1. In this paper, we focus on the more general weighted case; the method we present is also applicable to the unweighted Laplacian. Since the Fiedler vector can be computed independently for disconnected graphs, we assume that the graph is connected. The eigenvalues of $L_w$ are $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n$. The eigenvector $x_2$ corresponding to smallest nontrivial eigenvalue $\lambda_2$ is the sought Fiedler vector. If the matrix, $A$, is nonsymmetric we use $(|A| + |A^T|)/2$, instead.

A state of the art multilevel solver [7] called MC73Fiedler for computing the Fiedler vector is implemented in the Harwell Subroutine Library(HSL) [6]. It uses a series of levels of coarser graphs where the eigenvalue problem corresponding to the coarsest level is solved via the Lanczos method for estimating the
Fiedler vector. The results are then prolongated to the finer graphs and Rayleigh Quotient Iterations (RQI) with shift and invert are used for refining the eigenvector. Linear systems encountered in RQI are solved via the SYMMLQ algorithm. We consider MC73_Fiedler as one of the best uniprocessor implementation for determining the Fiedler vector.

In Section 2, we describe a novel parallel solver: TraceMin-Fiedler based on the Trace Minimization algorithm (TraceMin) [17, 16], and present results comparing it to MC73_Fiedler, in Section 3. Finally, in Section 4, we compare the quality of the Fiedler vectors obtained by both methods for reordering sparse matrices.

2 The TraceMin-Fiedler Algorithm

We consider solving the standard symmetric eigenvalue problem

\[ Lx = \lambda x \]  \hspace{1cm} (2)

where \( L \) denotes the weighted Laplacian, using the TraceMin scheme for obtaining the Fiedler vector. The basic TraceMin algorithm [17, 16] can be summarized as follows. Let \( X_k \) be an approximation of the eigenvectors corresponding to the \( p \) smallest eigenvalues such that \( X_k^T L X_k = \Sigma_k \) and \( X_k^T X_k = I \), where \( \Sigma_k = \text{diag}(\rho_1^{(k)}, \rho_2^{(k)}, ..., \rho_p^{(k)}) \). The updated approximation is obtained by solving the minimization problem

\[
\min \text{tr}(X_k - \Delta_k)^T L (X_k - \Delta_k), \quad \text{subject to } \Delta_k^T X_k = 0.
\]  \hspace{1cm} (3)

This in turn leads to the need for solving a saddle point problem, in each iteration of the TraceMin algorithm, of the form

\[
\begin{bmatrix}
L & X_k \\
X_k^T & 0
\end{bmatrix} \begin{bmatrix}
\Delta_k \\
N_k
\end{bmatrix} = \begin{bmatrix}
LX_k \\
0
\end{bmatrix}.
\]  \hspace{1cm} (4)

Where the Schur complement system \((X_k^T L^{-1} X_k)N_k = X_k^T X_k\) needs to be solved. Once \( \Delta_k \) and \( X_k \) are obtained \((X_k - \Delta_k)\) is then used to obtain \( X_{k+1} \) which forms the section

\[
X_{k+1}^T L X_{k+1} = \Sigma_{k+1}, X_{k+1}^T X_{k+1} = I.
\]  \hspace{1cm} (5)

The TraceMin-Fiedler algorithm, which is based on the basic TraceMin algorithm, is given in Figure 1.

The most time consuming part of the algorithm is solving the saddle-point problem in each outer TraceMin iteration. This involves, in turn, solving large sparse symmetric positive semi-definite systems of the form

\[ LW_k = X_k \]  \hspace{1cm} (6)

using the Conjugate Gradient algorithm with a diagonal preconditioner in Figure 2. Our main enhancement of the basic TraceMin scheme are contained in step 8, solving systems involving the Laplacian, and step 7 concerning the deflation process. In the TraceMin-Fiedler algorithm, not only is the coefficient matrix \( L \) is guaranteed to be symmetric positive semi-definite, but that its diagonal (the preconditioner) is guaranteed to have positive elements. On the other hand, in MC73_Fiedler there is no guarantee that the linear systems, arising in the RQI with shift and invert, are symmetric positive semi-definite with positive diagonal elements. Hence, MC73_Fiedler uses SYMMLQ without any preconditioning to solve linear systems in the Rayleigh Quotient Iterations.
Algorithm 1:

**Data:** $L$ is the $n \times n$ Laplacian matrix defined in Eqn. (1); $\varepsilon_{\text{out}}$ is the stopping criterion for the $||.||_{\infty}$ of the eigenvalue problem residual

**Result:** $x_2$ is the eigenvector corresponding to the second smallest eigenvalue of $L$

$p \gets 2; \quad q \gets 3p$

$n_{\text{conv}} \gets 0; \quad \mathbf{X}_{\text{conv}} \gets \begin{bmatrix} \end{bmatrix}$

$\hat{L} \gets L + ||L||_{\infty}10^{-12} \times \mathbf{I}$

$D \gets$ the diagonal of $L$

$\hat{D} \gets$ the diagonal of $\hat{L}$

$X_1 \gets \text{rand}(n,q)$

for $k = 1, 2, \ldots \max \text{it}$ do

1. Orthonormalize $X_k$ into $V_k$;

2. Compute the interaction matrix $H_k \gets V_k^T L V_k$;

3. Compute the eigendecomposition $H_k Y_k = Y_k \Sigma_k$ of $H_k$. The eigenvalues $\Sigma_k$ are arranged in ascending order and the eigenvectors are chosen to be orthogonal;

4. Compute the corresponding Ritz vectors $X_k \gets V_k Y_k$;

Note that $X_k$ is a section, i.e. $X_k^T L X_k = \Sigma_k, X_k^T X_k = \mathbf{I}$;

5. Compute the relative residual $||L X_k - X_k \Sigma_k||_{\infty}/||L||_{\infty}$;

6. Test for convergence: If the relative residual of an approximate eigenvector is less than $\varepsilon_{\text{out}}$, move that vector from $X_k$ to $X_{\text{conv}}$ and replace $n_{\text{conv}}$ by $n_{\text{conv}} + 1$ increment. If $n_{\text{conv}} \geq p$, stop;

7. Deflate: If $n_{\text{conv}} > 0, X_k \gets X_k - X_{\text{conv}} (X_{\text{conv}}^T X_k)$;

8. if $n_{\text{conv}} = 0$ then

    Solve the linear system $\hat{L} W_k = X_k$ approximately with relative residual $\varepsilon_{\text{in}}$ via the PCG scheme using the diagonal preconditioner $\hat{D}$;

else

    Solve the linear system $L W_k = X_k$ approximately with relative residual $\varepsilon_{\text{in}}$ via the PCG scheme using the diagonal preconditioner $D$;

9. Form the Schur complement $S_k \gets X_k^T W_k$;

10. Solve the linear system $S_k N_k = X_k^T X_k$ for $N_k$;

11. Update $X_{k+1} \gets X_k + \Delta_k = W_k N_k$

Figure 1: TraceMin-Fiedler algorithm.
Algorithm 2:

**Data:** \( \mathbf{L} \mathbf{x} = \mathbf{b} \), \( \mathbf{L} \) is the \( n \times n \) Laplacian matrix defined in Eqn. (1), \( \varepsilon_{in} \) is the stopping criterion for the \(||\cdot||_\infty\) of the relative residual, \( \mathbf{b} \) is the right hand side, and \( \mathbf{M} \) is the preconditioner

**Result:** \( \mathbf{x} \) is solution of the linear system

Solve the preconditioned system \((\mathbf{M}^{-1/2} \mathbf{L} \mathbf{M}^{-1/2})(\mathbf{M}^{1/2} \mathbf{x}) = (\mathbf{M}^{-1/2} \mathbf{b})\);

\[ \tilde{\mathbf{L}} = \mathbf{M}^{-1/2} \mathbf{L} \mathbf{M}^{-1/2} ; \]

\[ \tilde{\mathbf{b}} = \mathbf{M}^{-1/2} \mathbf{b} ; \]

\[ \tilde{x} = (\mathbf{M}^{1/2} \mathbf{x}) ; \]

\[ \tilde{\mathbf{r}}_0 \leftarrow [0, \ldots, 0]^T ; \]

\[ \tilde{\mathbf{p}}_0 \leftarrow \tilde{\mathbf{r}}_0 ; \]

for \( k = 1, 2, \ldots \) max it do

1. \( \alpha_k \leftarrow \frac{\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k}{\tilde{\mathbf{p}}_k^T \tilde{\mathbf{L}} \tilde{\mathbf{p}}_k} ; \)

2. \( \tilde{x}_{k+1} \leftarrow \tilde{x}_k + \alpha_k \tilde{\mathbf{p}}_k ; \)

3. \( \tilde{\mathbf{r}}_{k+1} \leftarrow \tilde{\mathbf{r}}_k - \alpha_k \tilde{\mathbf{L}} \tilde{\mathbf{p}}_k ; \)

4. if \(||\tilde{\mathbf{r}}_{k+1}||_\infty / ||\tilde{\mathbf{r}}_0||_\infty \leq \varepsilon_{in} \) then

   exit

5. \( \beta_k \leftarrow \frac{\tilde{\mathbf{r}}_{k+1}^T \tilde{\mathbf{r}}_{k+1}}{\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k} ; \)

6. \( \tilde{\mathbf{p}}_{k+1} \leftarrow \tilde{\mathbf{r}}_{k+1} + \beta_k \tilde{\mathbf{p}}_k ; \)

Figure 2: Preconditioned Conjugate Gradient Scheme for solving systems in the form \( \mathbf{L} \mathbf{x} = \mathbf{b} \).

We note, in TraceMin-Fiedler, that after the smallest eigenvector, which corresponds to the null space of \( \mathbf{L} \), has converged then in preconditioned CG in Figure 2,

\[ \tilde{\mathbf{p}}_k^T \tilde{\mathbf{L}} \tilde{\mathbf{p}}_k > 0. \]  

(7)

Observing that \( \mathbf{v} \perp \mathbf{b} \) due to the deflation step, the proof is given below.

**Theorem 2.1** Let \( \mathbf{L} \) be symmetric positive semidefinite such that \( \mathbf{L} \mathbf{v} = 0 \) (i.e. \( \mathcal{N}(\mathbf{L}) = \text{span}[\mathbf{v}] \)) and \( \mathbf{M} = \text{diag}(\mathbf{L}) \) ( \( \Rightarrow \mathbf{M}^{-1/2} \mathbf{L} \mathbf{M}^{-1/2} \mathbf{M}^{1/2} \mathbf{v} = 0 \Rightarrow \mathbf{L} \mathbf{v} = 0 \) where \( \mathbf{v} = \mathbf{M}^{1/2} \mathbf{v} \) )

The following statement is true for the preconditioned conjugate gradient method in Figure 2: if \( \mathbf{v} \perp \mathbf{b} \) then \( \tilde{\mathbf{v}} \perp \tilde{\mathbf{p}}_n \) and \( \tilde{\mathbf{v}} \perp \tilde{\mathbf{r}}_n \)

**Proof** (by induction)

- The base case:

\[
\tilde{\mathbf{v}}^T \tilde{\mathbf{p}}_0 = \tilde{\mathbf{v}}^T \tilde{\mathbf{r}}_0 = \tilde{\mathbf{v}}^T (\tilde{\mathbf{b}} - \tilde{\mathbf{L}} \tilde{\mathbf{x}}_0) (\text{note } x_0 = 0) = \tilde{\mathbf{v}}^T \tilde{\mathbf{b}} = \mathbf{v}^T \mathbf{M}^{1/2} \mathbf{M}^{-1/2} \mathbf{b} = \mathbf{v}^T \mathbf{b} = 0
\]
• Inductive hypothesis: Assume that $\tilde{v} \perp \tilde{p}_k$ and $\tilde{v} \perp \tilde{r}_k$ for $n = k$.

• Inductive step: Then for step $n = k + 1$,
  
  $$
  \tilde{v}^T \tilde{r}_{k+1} = \tilde{v}^T (\tilde{r}_k - \alpha_k \tilde{L} \tilde{p}_k)
  = 0 \quad \text{(by inductive hypothesis and } \tilde{v}^T \tilde{L} = 0)
  $$

  and

  $$
  \tilde{v}^T \tilde{p}_{k+1} = \tilde{v}^T (\tilde{r}_{k+1} + \beta_k \tilde{p}_k)
  = 0 + \beta_k \tilde{v}^T \tilde{p}_k
  = 0 \quad \text{(by inductive hypothesis)}
  $$

Therefore, we do not need to use a diagonal perturbation after the smallest eigenvalue and the corresponding eigenvector have converged.

We note that our algorithm can easily compute additional eigenvectors of the Laplacian matrix by setting $p$ to be the number of desired smallest eigenpairs.

Parallelism in the algorithm is achieved by partitioning all vectors, $X_k, V_k, W_k$ and the coefficient matrix $L$ by block rows where each MPI process contains one block. The matrix and the vectors are partitioned into blocks of roughly equal size. The most time consuming operation in Figure 1 is the solution of the linear systems involving $L$. The diagonal preconditioner does not require any communications. The sparse matrix-vector multiplications do require communication, however, with the amount of communication determined by the sparsity structure of the matrix. Therefore, the overall scalability of the algorithm is problem dependent. In the implementation in this paper we only communicate the elements that are needed to complete the product via asynchronous point to point communication (i.e. using MPI	exttt{ISEND} and MPI	exttt{IRECV}). The remaining operations that require communication are the inner products that use MPI	exttt{ALLREDUCE} with vectors of multiple columns.

3 Numerical Results

We implement the parallel TraceMin-Fiedler algorithm [12] in Figure 1 in parallel using MPI. We compare the parallel performance of MC73\_Fiedler with TraceMin-Fiedler using a cluster with Infiniband interconnection where each node consists of two quad-core Intel Xeon CPUs (X5560) running at 2.80GHz (8 cores per node). For both solvers we set the stopping tolerance for the $\infty$-norm of the eigenvalue problem residual to $10^{-5}$. In TraceMin-Fiedler we set the inner stopping criterion (relative residual norm for solving the linear systems using the preconditioned CG scheme) as $\varepsilon_{in} = 10^{-1} \varepsilon_{out}$, and the maximum number of the preconditioned CG iterations to be 30. For MC73\_Fiedler, we use all the default parameters.

The set of test matrices are obtained from the University of Florida (UF) Sparse Matrix Collection [2]. A search for matrices in this collection which are square, real, and which are of order $2,000,000 < N < 5,000,000$ returns the four matrices listed in Table 1. If the adjacency graph of $A$ has any disconnected single vertices, we remove them since those vertices are independent and have trivial solutions. We apply both MC73\_Fiedler and TraceMin-Fiedler to the weighted Laplacian generated from the adjacency graph of the preprocessed matrix where the weights are the absolute values of matrix entries. After obtaining the Fiedler vector $x_2$ produced by each algorithm, we compute the corresponding eigenvalue $\lambda_2$,

$$
\lambda_2 = \frac{x_2^T L x_2}{x_2^T x_2}.
$$

(8)
Table 1: Matrix size ($N$), number of nonzeros ($nnz$), and type of test matrices.

| Matrix Group/Name | $N$ | $nnz$ | symmetric | application         |
|-------------------|-----|-------|-----------|---------------------|
| 1. Rajat/rajat31  | 4,690,002 | 20,316,253 | no        | circuit simulation  |
| 2. Schenk/nlpkkt  | 3,542,400  | 95,117,792  | yes       | nonlinear optimization |
| 3. Freescale/Freescale1 | 3,428,755  | 17,052,626  | no        | circuit simulation  |
| 4. Zaoui/kktPower | 2,063,494  | 12,771,361  | yes       | optimum power flow  |

Table 2: Relative residuals $\|Lx - \hat{\lambda}x\|_{\infty}/\|L\|_{\infty}$ for TraceMin-Fiedler and MC73_Fiedler where $\varepsilon_{out} = 10^{-5}$.

| Matrix/Cores | TraceMin-Fiedler | MC73_Fiedler |
|--------------|-----------------|-------------|
| rajat31      | $1.1 \times 10^{-12}$ | $1.1 \times 10^{-12}$ | $1.1 \times 10^{-12}$ | $1.1 \times 10^{-12}$ | $3.03 \times 10^{-9}$ |
| nlpkkt       | $9.1 \times 10^{-6}$  | $9.1 \times 10^{-6}$  | $9.1 \times 10^{-6}$  | $9.1 \times 10^{-6}$  | $6.49 \times 10^{-7}$ |
| Freescale1   | $7.5 \times 10^{-12}$ | $7.5 \times 10^{-12}$ | $7.5 \times 10^{-12}$ | $7.5 \times 10^{-12}$ | $1.03 \times 10^{-7}$ |
| kktPower     | $3.1 \times 10^{-24}$ | $3.1 \times 10^{-24}$ | $3.1 \times 10^{-24}$ | $3.1 \times 10^{-24}$ | $4.07 \times 10^{-8}$ |

We report the relative residuals $\|Lx_2 - \hat{\lambda}_2x_2\|_{\infty}/\|L\|_{\infty}$ in Table 2.

The total time required by TraceMin-Fiedler using 1, 2, and 4 nodes with 8 MPI processes, i.e. using 8 cores, per node are presented in Table 3. We emphasize that the parallel scalability results for TraceMin-Fiedler is preliminary and that there is more room for improvement. Since MC73_Fiedler is purely sequential we have used it on a single core. The speed improvements realized by TraceMin-Fiedler on 1, 8, 16, and 32 cores over MC73_Fiedler on a single core are depicted in Figure 3 with the actual solve times and the speed improvement values are given in Tables 3 and 4. Note that on 32 cores, our scheme realizes speed improvements over MC73_Fiedler that range between 4 and 641 for our four test matrices.

Next, we compute the Fiedler vector of a symmetric matrix of dimension 11,333,520 $\times$ 11,333,520 and 61,026,416 nonzeros. The matrix is obtained from a 3D Finite Volume Method (FVM) discretization of a MEMS device. MC73_Fiedler consumes 75.5 seconds on a single core. The speed improvement of TraceMin-Fiedler is given in Table 4. We note that the results using single core on a node has a much more memory bandwidth available compared to 8 cores per node. Therefore, the speed improvement from 1 to 8 cores (all on a single node) is not ideal. TraceMin-Fiedler is 44.2 times faster than MC73_Fiedler using 256 cores.

Table 3: Total time in seconds (rounded to the first decimal place) for TraceMin-Fiedler and MC73_Fiedler and the average number of inner PCG iterations, number of outer TraceMin iterations for TraceMin-Fiedler.

| Matrix/Cores | TraceMin-Fiedler | MC73_Fiedler |
|--------------|-----------------|-------------|
| rajat31      | 2(1)            | 81.5s       |
| nlpkkt       | 2(30)           | 83.9s       |
| Freescale1   | 2(30)           | 52.8s       |
| kktPower     | 2(1)            | 341.6s      |
Figure 3: Speed improvement of TraceMin-Fiedler compared to uniprocessor MC73_Fiedler for four test problems.

Table 4: Speed improvement over MC73_Fiedler ($T_{MC73\_Fiedler}/T$).

| Matrix/Cores | TraceMin-Fiedler | MC73_Fiedler |
|--------------|-----------------|--------------|
| rajat31      | 14.5 59.2 116.5 227.5 | 1.0          |
| nlpkkt       | 0.8 3.4 5.5 7.8   | 1.0          |
| Freescale1   | 0.9 2.2 3.3 4.2   | 1.0          |
| kktPower     | 71.2 332.3 501.0 641.4 | 1.0          |
Using the Fiedler vector for permuting the elements of a matrix

One of the applications of the Fiedler vector is matrix reordering and bandwidth reduction. One can obtain the permutation to achieve reduction in the (weighted or nonweighted) bandwidth of the matrix by sorting the elements of the Fiedler vector (see [1] [10] for details).

In this section we propose a metric to measure the quality of the reordering, namely the relative bandweight. We compare the quality of the Fiedler vector using this metric.

We define the relative bandweight of a specified band of the matrix as follows:

$$w_k(A) = \frac{\sum_{i,j:|i-j|<k}|A(i,j)|}{\sum_{i,j}|A(i,j)|}.$$  

(9)

In other words, the bandweight of a matrix $A$, with respect to an integer $k$, is equal to the fraction of the total magnitude of entries that are encapsulated in a band of half-width $k$.

We randomly selected matrices with smaller dimension to be able to visualize the effect of reordering from the UF Sparse Matrix Collection in Table 5. The relative residuals for the Fiedler vector computed by both methods and the number of iterations for TraceMin-Fiedler is given in Table 6.

In 2 cases, namely bcsstk22 and cvxbqp1, out of 10, the relative residual of the Fiedler vector from MC73_Fiedler did not reach the stopping tolerance of $10^{-5}$. In Figures 5 and 12, we depict the relative bandweight comparison for these two cases and the resulting reordered matrices. In both cases TraceMin_Fiedler produces a better reordering. The relative residual of MC73_Fiedler ($3.5 \times 10^{-10}$) is significantly better than TraceMin_Fiedler ($2.3 \times 10^{-9}$) for sparsine. However, the quality of reordering is better for TraceMin_Fiedler using both our bandweight metric as well as the sparsity plots of the reordered
Table 5: Properties of test matrices.

| Matrix    | n   | nnz     | application         |
|-----------|-----|---------|---------------------|
| bcsstk22  | 138 | 696     | structural mechanics|
| problem1  | 414 | 2,779   | FEMLAB test matrix  |
| rail_1357 | 1,357 | 8,985 | heat transfer       |
| c-19      | 2,327 | 21,817 | nonlinear optimization|
| eurqsa    | 7,245 | 46,142 | economics           |
| tuma2     | 12,992 | 49,365 | mine model          |
| smt       | 25,710 | 3,749,582 | structural mechanics|
| cvxbqp1   | 50,000 | 349,968 | nonlinear optimization|
| sparsine  | 50,000 | 1,548,988 | structural optimization|
| F2        | 71,505 | 5,294,285 | structural mechanics|

matrices. For 6 cases out of 10, TraceMin_Fiedler generated a better reordering based on the sparsity plots and bandweights (see Figures 14, 13, 12, 11, 7, and 5), while in 3 cases (see Figures 10, 8, and 6) both methods produce comparable quality reorderings. Finally, for eurqsa, even though the bandweight measure indicates the reordering is slightly better if one uses MC73_Fiedler, the sparsity plots indicate better clustering of large elements using TraceMin_Fiedler.
Table 6: Relative residuals and the approximate eigenvalue($\lambda_2$).

| Matrix     | $\|L\|_\infty$ | TraceMin-Fiedler Relative Residual | $\lambda_2$ | # Outer(Avg. Inner) its. | MC73_Fiedler Relative Residual | $\lambda_2$ |
|------------|----------------|-----------------------------------|-------------|--------------------------|-------------------------------|-------------|
| bcsstk22   | $5.3 \times 10^6$ | $4.7 \times 10^{-6}$              | $6.0 \times 10^{-2}$ | 3(30)                    | $2.2 \times 10^{-5}$          | $2.8 \times 10^4$ |
| problem1   | $1.7 \times 10^1$  | $6.7 \times 10^{-6}$              | $4.6 \times 10^{-2}$ | 3(30)                    | $2.7 \times 10^{-6}$          | $4.6 \times 10^{-2}$ |
| rail1357   | $9.1 \times 10^{-5}$ | $8.2 \times 10^{-6}$              | $2.8 \times 10^{-9}$ | 4(30)                    | $5.4 \times 10^{-6}$          | $2.9 \times 10^{-8}$ |
| c-19       | $1.2 \times 10^4$  | $1.6 \times 10^{-6}$              | $3.8 \times 10^{-1}$ | 3(29)                    | $8.2 \times 10^{-6}$          | $4.0 \times 10^{-1}$ |
| eurqsa     | $1.3 \times 10^7$  | $5.3 \times 10^{-8}$              | $9.2 \times 10^{-1}$ | 2(30)                    | $2.9 \times 10^{-7}$          | $4.3 \times 10^{-1}$ |
| tuma2      | $1.0 \times 10^1$  | $2.6 \times 10^{-6}$              | $8.9 \times 10^{-4}$ | 8(30)                    | $9.5 \times 10^{-6}$          | $8.6 \times 10^{-4}$ |
| smt        | $1.8 \times 10^7$  | $8.3 \times 10^{-7}$              | $4.9 \times 10^{2}$  | 2(30)                    | $5.2 \times 10^{-6}$          | $2.0 \times 10^{-4}$ |
| cvxbqp1    | $7.0 \times 10^5$  | $6.2 \times 10^{-6}$              | $7.5 \times 10^{6}$  | 2(30)                    | $1.7 \times 10^{-2}$          | $9.4 \times 10^{3}$  |
| sparsine   | $3.2 \times 10^6$  | $2.3 \times 10^{-6}$              | $1.4 \times 10^{3}$  | 4(23)                    | $3.5 \times 10^{-10}$         | $1.0 \times 10^{5}$  |
| F2         | $4.2 \times 10^7$  | $1.5 \times 10^{-8}$              | $1.0 \times 10^{4}$  | 3(30)                    | $8.8 \times 10^{-6}$          | $4.7 \times 10^{2}$  |
Figure 5: Sparsity plots of *bcsstk22*: red and blue indicates the largest and the smallest elements, respectively, in the sparsity plots
Figure 6: Sparsity plots of *problem1*; red and blue indicates the largest and the smallest elements, respectively, in the sparsity plots.
Figure 7: Sparsity plots of rail_1357; red and blue indicates the largest and the smallest elements, respectively.
Figure 8: Sparsity plots of c-19; red and blue indicates the largest and the smallest elements, respectively.
Figure 9: Sparsity plots of *eurqsa*; red and blue indicates the largest and the smallest elements, respectively.
5 Conclusions

We have presented a new algorithm for computing the Fiedler vector on parallel computing platforms, and have shown its effectiveness compared to the well-known scheme given by routine MC73.Fiedler of the Harwell Subroutine Library for computing the Fiedler vector of four large sparse matrices. The scalability of the method was demonstrated for a matrix of dimension 11 million on a cluster. Finally, we have compared the quality of the reordering produced from the Fiedler vector for a variety of matrices from the UF sparse matrix collection and proposed the the bandweight as metric to measure the quality of the reordering.

Acknowledgments

The author is grateful to Professors Ahmed Sameh and Ananth Grama for their useful comments and supports, as well as introducing the idea of weighted reordering. The author also thanks Dr. Faisal Saied for providing an implementation of the trace minimization method and Eric Cox for his suggestions for improving the presentation of this paper.

References

[1] Stephen T. Barnard, Alex Pothen, and Horst Simon. A spectral algorithm for envelope reduction of sparse matrices. Numerical Linear Algebra with Applications, 2(4):317–334, 1995.

[2] T. A. Davis. University of Florida sparse matrix collection. NA Digest, 1997.

[3] M. Fiedler. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 23(2):298–305, 1973.

[4] Xiaofeng He, Hongyuan Zha, Chris H.Q. Ding, and Horst D. Simon. Web document clustering using hyperlink structures. Computational Statistics & Data Analysis, 41(1):19 – 45, 2002.

[5] Desmond J. Higham, Gabriela Kalna, and Milla Kibble. Spectral clustering and its use in bioinformatics. Journal of Computational and Applied Mathematics, 204(1):25 – 37, 2007. Special issue dedicated to Professor Shinnosuke Oharu on the occasion of his 65th birthday.

[6] HSL. A collection of Fortran codes for large-scale scientific computation, 2004. See http://www.cse.scitech.ac.uk/nag/hsl/.

[7] Y.F. Hu and J.A. Scott. HSL_MC73: a fast multilevel Fiedler and profile reduction code. Technical Report RAL-TR-2003-036, 2003.

[8] S. Kundu, D.C. Sorensen, and Jr. G.N. Philipshi. Automatic domain decomposition of proteins by a gaussian network model. Proteins: Structure, Function, and Bioinformatics, 57(4):725–733, 2004.

[9] M. Manguoglu. A parallel hybrid sparse linear system solver. In Computational Electromagnetics International Workshop, 2009. CEM 2009, pages 38–43, July 2009.

[10] M. Manguoglu, M. Koyuturk, A. Grama, and A. H. Sameh. Weighted matrix ordering and parallel banded preconditioners for iterative linear system solvers. SIAM Journal on Scientific Computing, accepted.
[11] Murat Manguoglu, Ahmed H. Sameh, and Olaf Schenk. Pspike: A parallel hybrid sparse linear system solver. *Lecture Notes in Computer Science (Euro-Par 2009 Parallel Processing)*, 5704:797–808, 2009.

[12] M. Manguoglu, F. Saied, E. Cox, and A. Sameh. http://www.cs.purdue.edu/homes/mmanguog/fiedler.html, 2010.

[13] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[14] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[15] Huaijun Qiu and Edwin R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22 – 34, 2006.

[16] Ahmed Sameh and Zhanye Tong. The trace minimization method for the symmetric generalized eigenvalue problem. *J. Comput. Appl. Math.*, 123(1-2):155–175, 2000.

[17] Ahmed H. Sameh and John A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 19(6):1243–1259, 1982.

[18] S. J. Shepherd, C. B. Beggs, and S. Jones. Amino acid partitioning using a fiedler vector model. *Journal European Biophysics Journal*, 37(1):105–109, 2007.
Figure 10: Sparsity plots of *tuma2*; red and blue indicates the largest and the smallest elements, respectively.
Figure 11: Sparsity plots of \textit{smt}; red and blue indicates the largest and the smallest elements, respectively, in the sparsity plots.
Figure 12: Sparsity plots of \textit{cvxbqp1}; red and blue indicates the largest and the smallest elements, respectively.
Figure 13: Sparsity plots of *sparsine*; red and blue indicates the largest and the smallest elements, respectively.
Figure 14: Sparsity plots of $F_2$; red and blue indicates the largest and the smallest elements, respectively.