TRACEMIN-Fiedler: A Parallel Algorithm for Computing the Fiedler Vector

Murat Manguoglu[1] and Eric Cox[1] and Faisal Saied[2] and Ahmed Sameh[1]

[1] Purdue University,
Department of Computer Science,
305 N. University Street,
West Lafayette, IN 47907
[2] Purdue University,
Computing Research Institute,
Room 202 250 North University Street,
West Lafayette, IN 47907

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. We compare the parallel performance of our method with a multilevel scheme, designed specifically for computing the Fiedler vector, which is implemented in routine MC73_FIEDLER of the Harwell Subroutine Library (HSL).

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 [10, 9, 8, 1], graph partitioning [12, 13], machine learning [11], protein analysis and data mining [5, 16], 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:

$$L_w(i,j) = \begin{cases} \sum_{\hat{j}} |A(i,\hat{j})| & \text{if } i = j, \\ -|A(i,j)| & \text{if } i \neq j. \end{cases} \tag{1}$$

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 $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_n$. The eigenvector x_2 corresponding to smallest nontrivial eigenvalue λ_2 is the sought Fiedler vector.

A state of the art multilevel solver [7] called MC73_FIEDLER 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 describe a novel parallel solver: TRACEMIN-Fiedler based on the Trace Minimization algorithm (TRACEMIN) [15, 14] in Section 2 and present results in Section 3 comparing it to MC73_FIEDLER.

2 The TRACEMIN-Fiedler Algorithm

We consider solving the standard symmetric eigenvalue problem

$$\mathbf{L}x = \lambda x \tag{2}$$

where L denotes the weighted Laplacian, using the TRACEMIN scheme for obtaining the Fiedler vector. The basic TRACEMIN algorithm [15, 14] can be summarized as follows. Let $\mathbf{X}_k$ be an approximation of the eigenvectors corresponding to the p smallest eigenvalues such that $\mathbf{X}_k^T \mathbf{L} \mathbf{X}_k = \Sigma_k$ and $\mathbf{X}_k^T \mathbf{X}_k = \mathbf{I}$, where $\Sigma_k = diag(\rho_1^{(k)}, \rho_2^{(k)}, ..., \rho_p^{(k)})$. The updated approximation is obtained by solving the minimization problem

$$\min \operatorname{tr}(\mathbf{X}_k - \Delta_k)^T \mathbf{L} (\mathbf{X}_k - \Delta_k), \quad \text{subject to } \Delta_k^T \mathbf{X}_k = 0. \tag{3}$$

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

$$\begin{bmatrix} \mathbf{L} & \mathbf{X}_k \\ \mathbf{X}_k^T & 0 \end{bmatrix} \begin{bmatrix} \Delta_k \\ \mathbf{N}_k \end{bmatrix} = \begin{bmatrix} \mathbf{L}\mathbf{X}_k \\ 0 \end{bmatrix}. \tag{4}$$

Once Δ_k is obtained $(\mathbf{X}_k - \Delta_k)$ is then used to obtain $\mathbf{X}_{k+1}$ which forms the section $\mathbf{X}_{k+1}^T \mathbf{L} \mathbf{X}_{k+1} = \Sigma_{k+1}, \mathbf{X}_{k+1}^T \mathbf{X}_{k+1} = \mathbf{I}$. The TRACEMIN-Fiedler algorithm, which based on the basic TRACEMIN algorithm, is given in Figure 1.

In step 4 the columns of the matrix $\mathbf{X}_k$ are orthonormal because columns of $\mathbf{V}_k$ and $\mathbf{Y}_k$ are orthonormal. 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 $\mathbf{L}\mathbf{W}_k = \mathbf{X}_k$ using the Conjugate Gradient algorithm with a diagonal preconditioner. 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

Algorithm 1:

Data: $\mathbf{L}$ is the $n \times n$ Laplacian matrix defined in Eqn.(1) , ε_{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 $\mathbf{L}$

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

$n_{conv} \longleftarrow 0; \quad \mathbf{X}_{conv} \longleftarrow [\quad]$;

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

$\mathbf{D} \longleftarrow$ the diagonal of $\mathbf{L}$;

$\hat{\mathbf{D}} \longleftarrow$ the diagonal of $\hat{\mathbf{L}}$;

$\mathbf{X}_1 \longleftarrow rand(n, q)$;

for $k = 1, 2, \ldots$ *max_it* **do**

1. Orthonormalize $\mathbf{X}_k$ into $\mathbf{V}_k$;
2. Compute the interaction matrix $\mathbf{H}_k \longleftarrow \mathbf{V}_k^T \mathbf{L} \mathbf{V}_k$;
3. Compute the eigendecomposition $\mathbf{H}_k \mathbf{Y}_k = \mathbf{Y}_k \Sigma_k$ of $\mathbf{H}_k$. The eigenvalues Σ_k are arranged in ascending order and the eigenvectors are chosen to be orthogonal;
4. Compute the corresponding Ritz vectors $\mathbf{X}_k \longleftarrow \mathbf{V}_k \mathbf{Y}_k$;
Note that $\mathbf{X}_k$ is a section, i.e. $\mathbf{X}_k^T \mathbf{L} \mathbf{X}_k = \Sigma_k, \mathbf{X}_k^T \mathbf{X}_k = \mathbf{I}$;
5. Compute the relative residual $||\mathbf{L}\mathbf{X}_k - \mathbf{X}_k \Sigma_k||_\infty / ||\mathbf{L}||_\infty$;
6. Test for convergence: If the relative residual of an approximate eigenvector is less than ε_{out}, move that vector from $\mathbf{X}_k$ to $\mathbf{X}_{conv}$ and replace n_{conv} by $n_{conv} + 1$ increment. If $n_{conv} \geq p$, stop;
7. Deflate: If $n_{conv} > 1, \mathbf{X}_k \longleftarrow \mathbf{X}_k - \mathbf{X}_{conv}(\mathbf{X}_{conv}^T \mathbf{X}_k)$;
8. **if** $k = 1$ **then**
   Solve the linear system $\hat{\mathbf{L}} \mathbf{W}_k = \mathbf{X}_k$ approximately via the PCG scheme using the diagonal preconditioner $\hat{\mathbf{D}}$;
   else
   Solve the linear system $\mathbf{L} \mathbf{W}_k = \mathbf{X}_k$ approximately via the PCG scheme using the diagonal preconditioner $\mathbf{D}$;
9. Form the Schur complement $\mathbf{S}_k \longleftarrow \mathbf{X}_k^T \mathbf{W}_k$;
10. Solve the linear system $\mathbf{S}_k \mathbf{N}_k = \mathbf{X}_k^T \mathbf{X}_k$ for $\mathbf{N}_k$ directly;
11. Update $\mathbf{X}_{k+1} \longleftarrow \mathbf{X}_k - \Delta_k = \mathbf{W}_k \mathbf{N}_k$;

Fig. 1. TRACEMIN-Fiedler algorithm.

positive semi-definite with positive diagonal elements. Hence, MC73_FIEDLER uses SYMMLQ without any preconditioning to solve linear systems in the Rayleigh Quotient Iterations.

We should note here that the matrix $\mathbf{L}$ is symmetric positive semi-definite with one zero eigenvalue. As soon as the first eigenvalue has converged, however, the right hand side $\mathbf{X}_k$ is orthogonal to the null space of $\mathbf{L}$ due to the deflation step 7. Since the smallest (i.e. 0) eigenvalue converges after the first iteration of the algorithm we add a small diagonal perturbation for the first iteration of the algorithm only in order to ensure PCG will not fail.

The order of the linear system in step 10 is $q \times q$ where $q = 6$, therefore we solve these small systems directly. We note that our algorithm can easily compute additional eigenvectors of the Laplacian matrix by setting p to be the number of desired of smallest eigenpairs.

3 Parallel Implementation of TRACEMIN-Fiedler

The parallel TRACEMIN-Fiedler algorithm consists of the same basic steps as the serial algorithm 1. The matrix and vectors are distributed in blocks across the processors. Our parallel implementation is based on the MPI communication library.

One critical part of the parallelization is the matrix vector product. Due to the block nature of the TRACEMIN algorithm, the matrix L is applied to a set of vectors at a time, which leads to greater efficiency. The amount of communication needed in the matrix vector product is problem dependent. The scalability of this operation and therefore of the overall parallel TRACEMIN-Fiedler algorithm varies depending on the number of non-zeros in L and their location. The parallel matrix-vector multiplication operation is performed in Step 2, for the computation of $\mathbf{H}_k$, in Step 5, for computing the residuals, and once in each iteration of the PCG solve in Step 8.

The other type of communication needed in the parallel TRACEMIN-Fiedler algorithm is the AllReduce operation. This is required in the computation of dot products and norms. In particular, the AllReduce communication is performed in Step 1, for the orthonormalization step, in Step 2, for the computation of $\mathbf{H}_k$, on Step 5, in the computation of the residual norms, in Step 7, for the deflation operation and in Step 9, in the computation of the Schur complement matrix. The AllReduce communication operation is performed three times in each iteration of the PCG solve in Step 8. In our implementation, most AllReduce operations are applied to a set of vectors, which is more efficient than doing more reductions one at a time.

4 Numerical Results

We implement the TRACEMIN-Fiedler algorithm 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 as $\varepsilon_{in} = 10^{-1} * \varepsilon_{out}$, and the maximum number of the preconditioned CG to 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 four matrices listed in Table 1. If a matrix, A, is nonsymmetric we use $(|A| + |A^T|)/2$, instead. Furthermore, if the adjacency graph of A has any disconnected single vertices we removed 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 returned by each algorithm, we compute the corresponding eigenvalue λ_2,

$$\lambda_2 = \frac{x_2^T L x_2}{x_2^T x_2}. \tag{5}$$

We report the relative residuals $||Lx_2 - \lambda_2 x_2||_\infty / ||L||_\infty$ in Table 2.

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

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

Table 2. Relative residuals $\|Lx - \lambda x\|_\infty / \|L\|_\infty$ for TRACEMIN-Fiedler and MC73_FIEDLER where $\varepsilon_{out} = 10^{-5}$.

	TRACEMIN-Fiedler				MC73_FIEDLER
Matrix/Cores	1	8	16	32	1
rajat31	1.1×10^{-12}	1.1×10^{-12}	1.1×10^{-12}	1.1×10^{-12}	3.03×10^{-9}
nlpkkt	9.1×10^{-6}	9.1×10^{-6}	9.1×10^{-6}	9.1×10^{-6}	6.49×10^{-7}
Freescale1	7.5×10^{-12}	7.5×10^{-12}	7.5×10^{-12}	7.5×10^{-12}	1.03×10^{-7}
kktPower	3.1×10^{-24}	3.1×10^{-24}	3.1×10^{-24}	3.1×10^{-24}	4.07×10^{-8}

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 2, with the

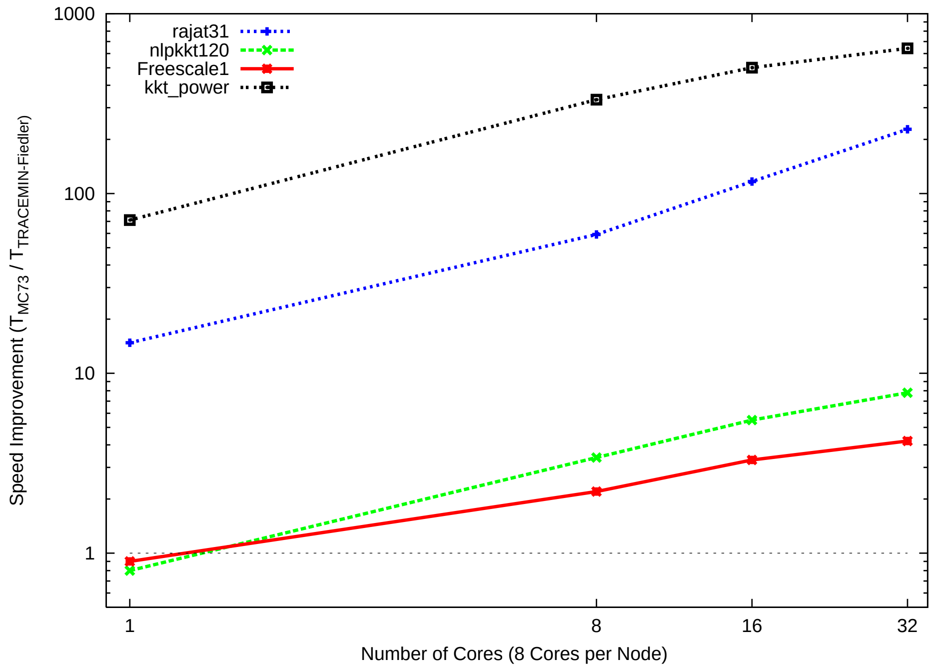

Fig. 2. Speed improvement of TRACEMIN-Fiedler compared to uniprocessor HSL_MC73 for four test problems.

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.

Table 3. Total time in seconds (rounded to the first decimal place) for TRACEMIN-Fiedler and MC73_FIEDLER.

	TRACEMIN-Fiedler				MC73_FIEDLER
Matrix/Cores	1	8	16	32	1
rajat31	5.6	1.4	0.7	0.4	81.5
nlpkkt	100.5	24.9	15.3	10.8	83.9
Freescale1	61.5	23.5	16.0	12.5	52.8
kktPower	4.8	1.0	0.7	0.5	341.6

Table 4. Speed improvement over MC73_FIEDLER ($T_{MC73\_FIEDLER}/T$).

	TRACEMIN-Fiedler				MC73_FIEDLER
Matrix/Cores	1	8	16	32	1
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

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.

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. M. Manguoglu. A parallel hybrid sparse linear system solver. In *Computational Electromagnetics International Workshop, 2009. CEM 2009*, pages 38–43, July 2009.
9. 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.
10. 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.
11. 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.
12. 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.
13. Huaijun Qiu and Edwin R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22 – 34, 2006.
14. 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.
15. 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.
16. 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.