# Monte Carlo Techniques for Estimating the Fiedler Vector in Graph Applications

Ashok Srinivasan and Michael Mascagni

Department of Computer Science, Florida State University,
Tallahassee FL 32306, USA
{asriniva, mascagni}@cs.fsu.edu

**Abstract.** Determining the Fiedler vector of the Laplacian or adjacency matrices of graphs is the most computationally intensive component of several applications, such as graph partitioning, graph coloring, envelope reduction, and seriation. Often an approximation of the Fiedler vector is sufficient. We discuss issues involved in the use of Monte Carlo techniques for this purpose.

## 1 Introduction

The Fiedler vector is the eigenvector corresponding to the second smallest non-negative eigenvalue of a matrix. Spectral techniques, based on determining the Fiedler vector of the Laplacian or the adjacency matrix of a graph, are used in graph partitioning [18], matrix envelope reduction [3, 9], seriation [2], and graph coloring algorithms [1]. We introduce terminology and the background for this problem in § 2.

Applications of the above algorithms arise in diverse areas, such as parallel computing, VLSI, molecular dynamics, DNA sequencing, databases, clustering, linear programming, scheduling, and archaeological dating [1, 2, 5, 8, 10, 11, 16, 18]. (A computationally analogous problem, that of determining the second largest eigenvalue, has important applications in statistical and quantum mechanics as well [17].) We outline some applications in § 2.1 and § 2.2.

While spectral techniques for the algorithms mentioned above often give results of high quality, a major computational bottleneck is the determination of the Fiedler vector, which has often limited the application of spectral techniques. We observe that in most of the applications mentioned above, only an approximation to the Fiedler vector is required. Our work is directed at reducing the computational effort through the use of Monte Carlo (MC) techniques. We review MC techniques for computing eigenvalues and eigenvectors in § 3, and in § 4 we discuss ideas for using the properties of graphs to speed up the computations.

We present preliminary experimental results on the use of MC techniques to graph application in § 5. We conclude by suggesting directions for further research, in § 6.

## 2 Background

The eigenvectors of the adjacency matrix and the Laplacian of a graph have important applications. The latter has especially been popular in recent times, and so we shall focus on it. It can be shown that all the eigenvalues of the Laplacian, $L$ are non-negative, and exactly one of them is 0 for a connected graph. The eigenvector $e_2$ corresponding to the second smallest eigenvalue, $\lambda_2$, is called its *Fiedler vector*, and $\lambda_2$ is referred to as the *Fiedler value*. In the techniques mentioned in § 1, typically the Fiedler vector is determined and the vertices of the graph are characterized by appealing to the corresponding components of the Fiedler vector, as shown below for graph partitioning and seriation.

### 2.1 Graph Partitioning

Given a graph $G = (V, E)$ with $n$ vertices, and number of partitions $P$, the most popular graph partitioning problem partitions the set of vertices $V$ into $P$ disjoint subsets such that each subset has equal size and the number of edges between vertices in different subsets is minimized[1]. Graph partitioning has applications in parallel computing, VLSI, databases, clustering, linear programming, and matrix reordering. This problem is NP-hard, and therefore various heuristics are used to find a good approximation.

The spectral heuristic determines the Fiedler vector of the Laplacian, and components smaller than the median are placed in one subset, say $V_1$, and the rest in subset $V_2$. Each of these subsets is recursively partitioned using the spectral method. The spectral method typically gives good quality partitions, but requires enormous computational effort for large graphs. Therefore other methods based on multilevel techniques have become popular, while there has been a simultaneous effort to improve the speed of the spectral method [14, 20].

### 2.2 Seriation

Consider a set of elements $1, 2, \ldots, n$ and a similarity function $f$. The seriation problem [2] problem is to determine a permutation $\pi$ such that $\pi(i) < \pi(j) < \pi(k) \Rightarrow f(i, j) \geq f(i, k)$. Intuitively, a high value of $f(i, j)$ indicates a strong desire for elements $i$ and $j$ to be near each other, and through the permutation $\pi$ we order the elements so that they satisfy the requirements of the function $f$. $f$ may be such that it is impossible to find a permutation satisfying it; in that case, minimization of the following penalty function [2] can be attempted: $g(\pi) = \sum_{i,j} f(i, j)(\pi_i - \pi_j)^2$. This minimization is NP-hard, and so approximation techniques are used.

In the spectral method for this problem, a continuous version of the above penalty function minimization is solved. This leads to the following computational technique [2]: The Fiedler vector of the matrix corresponding to the similarity function $f$ is determined. Then the elements are ordered based on the

[1] Metrics, other than the number of edges between different partitions, are also used in certain applications.

magnitude of the corresponding component of the Fiedler vector. Applications of this technique arise in DNA sequencing, archaeological dating, and also in a closely related problem of matrix envelope reduction.

## 3 Monte Carlo for Eigenvector Computations

We note that MC techniques are the ideal algorithms to use when approximate solutions are sufficient. MC computations of eigenvalues and eigenvectors are typically stochastic implementations of the power method [17].

### 3.1 Power Method

We now summarize the MC algorithm to compute the eigenvector for the eigenvalue of largest magnitude, based on the algorithm given in [6, 7]. Consider a matrix $A \in \Re^{n\times n}$ and a vector $h \in \Re^n$. An eigenvector corresponding to the largest eigenvalue can be obtained through the power method as $\lim_{i\to\infty} A^i h$ for a starting vector $h$ that has a non-zero component in the direction of the desired eigenvector. The MC techniques is based on estimating $A^m h$ using Markov chains of length $m$, for large $m$. The random walk visits a set of states in $\{1, \ldots, n\}$. Let the state visited in the $i$ th step be denoted by $k_i$. Then the probability that the start state $k_0$ is $\alpha$ is given by

$$P_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^{n} |h_\alpha|}, \tag{1}$$

and the transition probability is given by

$$Pr(k_i = \alpha | k_{i-1} = \beta) = P_{\alpha\beta} = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|}. \tag{2}$$

Consider random variables $W_i$ defined as follows

$$W_0 = \frac{h_{k_0}}{P_{k_0}},\ W_i = W_{i-1}\frac{a_{k_i k_{i-1}}}{p_{k_i k_{i-1}}}. \tag{3}$$

If we let $\delta$ denote the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$, and 0 otherwise), then it can been shown [7] that

$$E(W_i \delta_{\alpha k_i}) = (A^i h)_\alpha. \tag{4}$$

Therefore, in order to implement the power method, we evaluate $E(W_i)$ for large $i$ to estimate the $k_i$ th component of the largest eigenvector of $A$.

### 3.2 Inverse Power Method

If we need to determine an eigenvector for the eigenvalue of smallest magnitude, then we can apply a stochastic version of the inverse power method, where the desired eigenvector is computed as $\lim_{i\to\infty}(A^{-1})^i h$ for a starting vector $h$ that has a non-zero component in the direction of the desired eigenvector. In the deterministic simulation, we repeatedly solve the following linear system

$$Ax_{k+1} = x_k,\ x_0 = h. \tag{5}$$

In the MC technique, we replace the deterministic linear solver with a MC solver. There are several techniques for solving a linear system through MC. The basic idea is to write $A$ a as $I-C$, and conceptually use a MC version of the stationary iteration

$$y_k = Cy_{k-1} + h = \sum_{i=0}^{k-1} C^i y_0,\ y_0 = h, \tag{6}$$

which converges to the the desired solution if the spectral radius of $C$ is less than 1. We can use the MC technique for computing Matrix-vector products, described in Sec 3.1, to determine each $C^i y_0$ on the right hand side of Eqn. (6). This corresponds to the MC estimator introduced by Wasow [19].

## 4 Improving the Convergence of Monte Carlo Techniques

We discuss ideas to accelerate the convergence of the MC techniques for the graph applications, in this section.

### 4.1 Deflation:

The MC techniques are essentially the power or inverse iteration methods, and therefore, they too retrieve the extreme eigenvalues and eigenvectors. Since we wish to obtain the second smallest one, we deflate the matrix analytically. This can be done for both the adjacency matrix and the Laplacian. We demonstrate it for the Laplacian, since its use is more popular in recent times.

The smallest eigenvalue of the Laplacian is given by $\lambda_1 = 0$ with the corresponding eigenvector $e_1 = (1, 1, \ldots, 1)^T$. We wish to find a non-singular matrix $H$ (See Heath [12], page 127) such that

$$He_1 = \alpha(1, 0, 0, \ldots, 0)^T, \tag{7}$$

for some constant $\alpha$. Then

$$HLH^{-1} = \begin{bmatrix} \lambda_1 & b^T \\ 0 & B \end{bmatrix}, \tag{8}$$

where $B$ has eigenvalues $\lambda_2, \lambda_3, \ldots, \lambda_n$. Furthermore, if $y_2$ is the eigenvector corresponding to eigenvalue $\lambda_2$ in $B$, then

$$e_2 = H^{-1} \begin{bmatrix} \alpha \\ y_2 \end{bmatrix},\ \text{where } \alpha = \frac{b^T y_2}{\lambda_2 - \lambda_1}. \tag{9}$$

(This requires $\lambda_2 \neq \lambda_1$, which is satisfied due to our assumption that the graph is connected.)

Of course, we wish to find an $H$ such that $B$ and $e_2$ can be computed fast. Furthermore, since most applications will have sparse graphs, we wish to have $B$ preserve the sparsity. Consider $H$ defined as follows:

$$h_{ij} = \begin{cases} -1, \text{j} = 1 \\ 1, i = j, \text{and } i > 1 \\ 0, \text{otherwise} \end{cases}. \tag{10}$$

Then $H$ satisfies Eqn. (7) with $\alpha = -1$. Furthermore, $H^{-1} = H$ and we can show that in Eqn. (8)

$$b_{ij} = l_{i+1,j+1} - l_{1,j+1},\ 1 \leq i, j \leq n-1, \tag{11}$$

$$b_i = -l_{1,i+1},\ 1 \leq i \leq n-1. \tag{12}$$

We can verify that $B$ too is sparse. The number of non-zero elements in $L$ is $\sum_{i=1}^{n} d_i + n$, where $d_i$ is the degree of vertex $i$. Let $V_1$ be the set of nodes to which vertex 1 does not have an edge (excluding vertex 1 itself), and $V_2$ be the set of vertices to which vertex 1 has an edge. The number of non-zero elements in column $i-1$ of $B$ is $d_i + 1$ if $i \in V_1$, and the number of non-zero entries is $n - 1 - d_i$ if $i \in V_2$ (since the $j$ th row entry of $B$ will be non-zero unless there is an edge between $i$ and $j$, when $i \in V_2$). Therefore the total number of non-zero entries is given by:

$$\sum_{i \in V_1} (d_i + 1) + \sum_{i \in V_2} (n - 1 - d_i) \tag{13}$$

$$= \sum_{i \in V_1} d_i - \sum_{i \in V_2} d_i + (n-2)d_1 + n - 1 \tag{14}$$

$$< \sum_{i=2}^{n} d_i + (n-1)d_1 + n - 1. \tag{15}$$

If we label the vertices so that the vertex with least degree is labeled 1, then from Eqn. (15) we can see that an upper bound on the number of non-zero entries of $B$ is given by

$$2\sum_{i=2}^{n} d_i + n - 1. \tag{16}$$

Therefore the sparsity of $B$ is decreased by at most a factor of 2 compared with just considering the the submatrix of $L$ that omits the first row and column.

We also need to determine $e_2$ from $y_2$. We observe that with our definition of $H$, we get

$$e_2 = \begin{bmatrix} -\alpha \\ y_2 - \boldsymbol{\alpha} \end{bmatrix}, \text{ where } \alpha = \frac{b^T y_2}{\lambda_2}, \text{ and } \boldsymbol{\alpha} = (\alpha, \ldots, \alpha)^T. \tag{17}$$

In fact, if only the relative values of the components is important, as in the applications mentioned in § 1, then we need not compute $e_2$ explicitly, and can instead use the vector $(0\ y_2^T)^T$.

### 4.2 Shifting:

We can apply inverse iterations to the matrix $B$ above, both in deterministic and MC techniques, in order to estimate $\lambda_2$, and, more importantly for our purpose, $y_2$. Alternatively, since solving linear systems repeatedly is expensive, we can shift $B$ to enable computation of the desired solution as the eigenvector corresponding to the eigenvalue largest in magnitude. For example, the largest eigenvalue is bounded by $\Delta = 2\sum_{i=1}^{n} d_i$. Therefore we can apply the power method based MC technique to $\Delta I - B$ and obtain the eigenvector corresponding to eigenvalue $\Delta - \lambda_2$.

## 5 Experimental Results

In this section, we discuss the experimental set up, implementation details, and results of deterministic and MC solutions. We chose the graph partitioning application to test the techniques described above, and used two sample graphs for our tests – *test.graph* from Metis [15] and *hammond.graph* from Chaco [13]. The former is a small graph with 766 vertices and 1314 edges, while the latter, a 2D finite element grid of a complex airfoil with triangular elements, is of moderate size, having 4720 vertices and 13722 edges. All the coding was done on Matlab, and executed on a 1.0GHz Pentium III running Linux. We tested the effectiveness of techniques by partitioning the graphs into two disjoint parts. The ratio of the number of edges cut using our techniques to the number of edges cut using the exact Fiedler vector gives a measure of the effectiveness, with a large number indicating poor performance, and a small one good performance.

Since many of the techniques, such as deflation and shifting, apply to deterministic as well as MC techniques, we first present results demonstrating the effectiveness in deterministic techniques. We first computed the eigenvectors of the two largest eigenvalues of the sparse matrix $\Delta I - L$, using the *eigs* routine in Matlab. The time taken and number of edges cut using the Fiedler vector obtained from this step were used to measure the effectiveness of our techniques. In the rest of the tests, we divided the $L$ matrix by $\Delta$ in order for all the eigenvalues of its deflation to be in $(0, 1]$.

We next tested the effectiveness of the deflation technique mentioned in § 4.1 with the deterministic inverse iterations. We could have started out with a random vector for the inverse iterations. However, a random initial vector would have a component in the direction of the the eigenvector for the 0 eigenvalue, which is not conducive to fast convergence. Hence we chose the initial start vector by assigning the first $n/2$ vertices of the graph to the same partition, and the rest to the other one, and then applied the deflation process to get a starting vector for the iterations. We can see from Fig. 1 that this starting vector is more effective than using a random vector, since our scheme leads to identical results

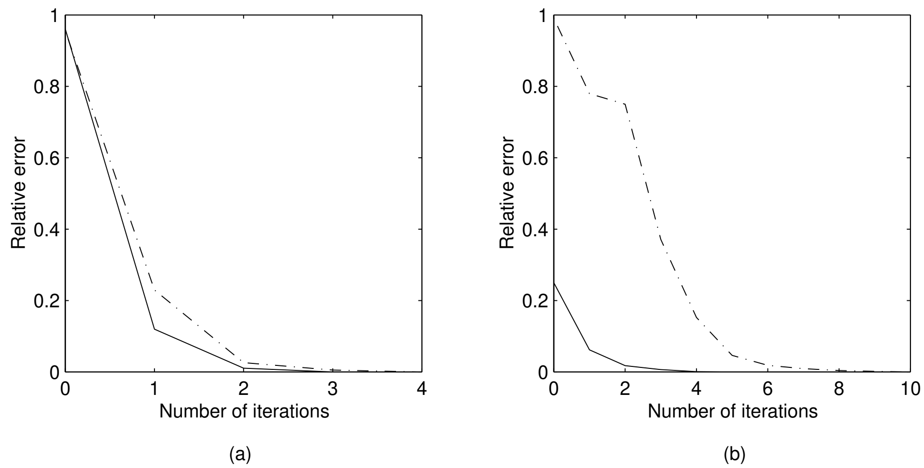



**Fig. 1.** Comparison of the effectiveness of using a random start vector versus our scheme. The solid line shows the relative error for our scheme, while the dash-dotted line shows the relative error for a random start vector with (a) *test.graph* and (b) *hammond.graph*.

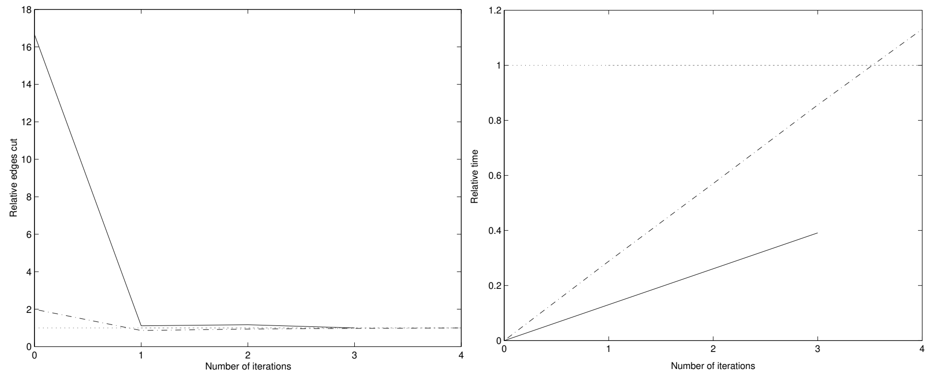
with that of the exact Fiedler vector in fewer number of iterations of the linear solver[2].

We can see from Fig. 2 that deterministic inverse iterations give good results with just one linear solve, and agree with the exact Fiedler vector with just a few linear solves. Furthermore, the timing result indicates significant improvement over the direct technique.

We next show in Fig. 3 that deterministic power method applied to $I - B/\Delta$ too is effective. While the number of iterations required is very high, the time taken is low, since matrix-vector multiplication is much faster than solving a linear system. However, note that in the corresponding MC scheme, this would imply that each walk has to have a few hundred steps, and therefore the MC version of this method would not be effective. We therefore restrict our attention to the inverse iteration scheme for MC.

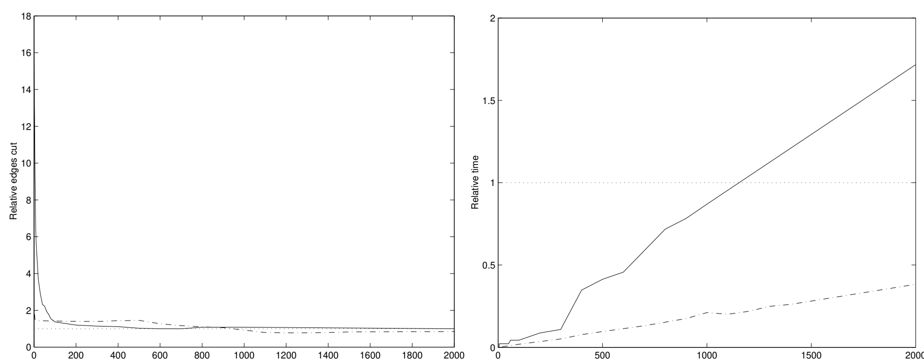
We show results for *test.graph* with MC inverse iterations in Fig. 4. The walk length was fixed at 5 for graphs $a, b, c$, and plots of edges cut versus number of iterations of the linear solver are presented, for different numbers of simulations. We make the following observations (i) as expected, the accuracy increases with increase in the number of simulations per linear solve, (ii) in terms of time taken (note, 100 iterations with 10 simulations each takes the same time as 1 iteration with 1000 simulations each), it may be useful to use fewer simulations per iteration in return for a larger number of linear solves, and (iii) the results

[2] The relative error here is the ratio of number of vertices in a partition inconsistent with that of the exact Fiedler vector, to the maximum that can be in the "wrong" partition.




(a) (b)

**Fig. 2.** Plot of edges cut and time using the deflated matrix, relative to that of the exact Fiedler vector, for different numbers of inverse iterations. The solid line shows the plot for *test.graph*, while the dash-dotted line plots the curve for *hammond.graph*.




(a) (b)

**Fig. 3.** Plot of edges cut and time using the deflation matrix, relative to that of the exact Fiedler vector, for different numbers of power method iterations. The solid line shows the plot for *test.graph*, while the dash-dotted line plots the curve for *hammond.graph*.

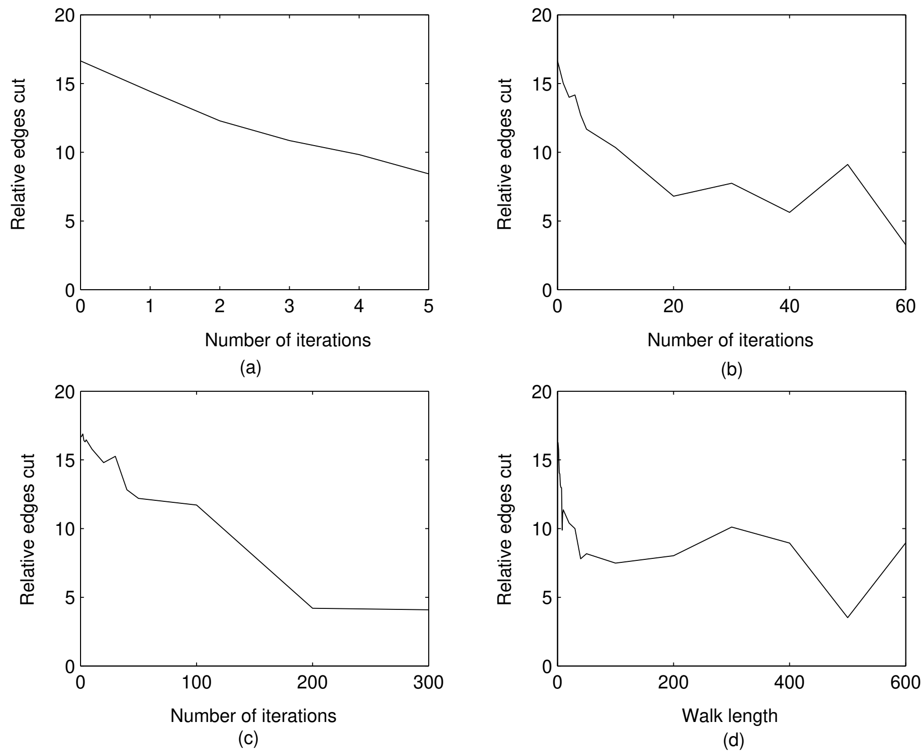



**Fig. 4.** Plot of edges cut using the MC inverse iterations. Graphs $a, b, c$ use a walk length of 5 and number of iterations of 1000 (a), 100 (b), and 10 (c) respectively. Graph (d) keeps the number of iterations fixed at 3 and number of simulations per iteration at 100.

from the MC technique are not good enough. Graph (d) has varying walk length, holding the number of simulations fixed at 100 and number of iterations of the linear solver at 3.

A reason for the inadequate performance of the MC scheme is that the matrix $C$ of Eqn. 6 has spectral radius close to 1 in our application. This can be improved by changing the stationary iteration scheme. For example, Jacobi iterations can easily be incorporated into the current MC framework. Fig. 5 shows that the relative error[3] in a single linear solve using (deterministic) Jacobi iterations can be much smaller than with the process we have used. However, we need to prove convergence of these methods for the graph under consideration, before using them. Furthermore, the MC methods corresponding to the better stationary schemes, such as Gauss-Seidel and SOR are yet to be developed. Meanwhile, hybrid techniques combining MC for the matrix-vector multiplication and de-

[3] The relative error here is $\|x - \hat{x}\|/\|x\|$, where $x$ is the exact solution and $\hat{x}$ is the MC estimate.

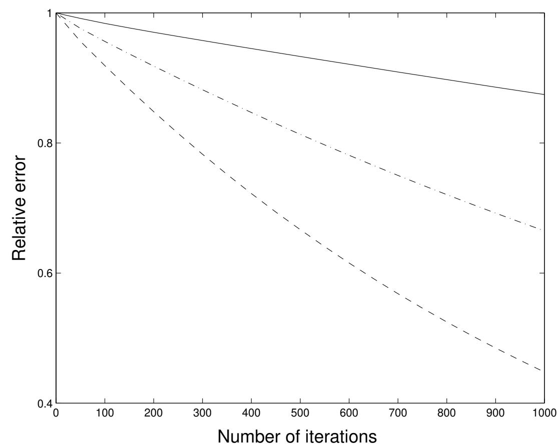



**Fig. 5.** Plot of relative error in a single linear solve using the current stationary process (solid line), Jacobi iterations (dash-dotted line), and Gauss-Seidel (dashed line). The deflated, scaled Laplacian of *test.graph* was used for this test.

terministic techniques for the linear solve involved in these two stationary techniques could be used. Development of MC versions of non-stationary methods is an open problem that could lead to more effective solutions for this problem.

## 6 Conclusions

We have outlined ideas for MC to be used in graph partitioning and related applications that require the estimation of the Fiedler vector of the graph Laplacian. We suggested techniques for improvement in convergence, for both deterministic and MC techniques. We demonstrated their effectiveness in deterministic calculations. However, the results with MC are not sufficiently good. We have identified the problem – the type of stationary process used is not suitable for graph applications, and suggested further research into the use of MC linear solvers that use better stationary schemes, or even non-stationary ones. Yet another research issue is to consider variance reduction techniques, for example, based on approximations to the Fiedler vector obtained using multilevel techniques as in [4].

## References


1. B. Aspvall and J. R. Gilbert. Graph coloring using eigenvalue decomposition. *SIAM J. Alg. Disc. Meth.*, 5:526–538, 1984.
2. J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28:297–310, 1998.
3. S. T. Barnard, A. Pothen, and H. D. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numer. Linear Algebra Appl.*, 2:317–334, 1995.
4. S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–107, 1994.
5. Shashi Shekhar Chang-Tien. Optimizing join index based spatial-join processing: A graph partitioning approach.
6. I. T. Dimov and A. N. Karaivanova. Parallel computations of eigenvalues based on a Monte Carlo approach. *Monte Carlo Methods and Applications*, 4:33–52, 1998.
7. I. T. Dimov and A. N. Karaivanova. A power method with monte carlo iterations. In Iliev, Kaschiev, Margenov, Sendov, and Vassilevski, editors, *Recent Advances in Numerical Methods and Appl. II*, pages 239–247. World Scientific, 1999.
8. B. E. Eichinger. Configuration statistics of Gaussian molecules. *Macromolecules*, 13:1–11, 1980.
9. A. George and A. Pothen. An analysis of spectral envelope reduction via quadratic assignment problems. *SIAM Journal on Matrix Analysis and Applications*, 18:706–732, 1997.
10. D. S. Greenberg and S. Istrail. Physical mapping by STS hybridization: algorithmic stratgies and the challenge of software evaluation. *J. Comp. Biol.*, 2:219–273, 1995.
11. L. Hagen and A. Kahng. Fast spectral methods for ratio cut partitioning and clustering, 1991.
12. M. T. Heath. *Scientific Computing: An introductory survey.* McGraw-Hill, New York, 1997.
13. B. Hendrickson and R. Leland. The Chaco user's guide — version 2.0, 1994.
14. M. Holzrichter and S. Oliveira. A graph based davidson algorithm for the graph partitioning problem. *IJFCS: International Journal of Foundations of Computer Science*, 10:225–246, 1999.
15. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, 1995.
16. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–92, 1998.
17. M. P. Nightingale and C. J. Umrigar. Monte Carlo eigenvalue methods in quantum mechanics and statistical mechanics. In D. M. Ferguson, J. I. Siepmann, and D. G. Truhlar, editors, *Monte Carlo Methods in Chemical Physics*, volume 105 of *Advances in Chemical Physics*, chapter 4. John Wiley and Sons, New York, 1999.
18. A. Pothen, H. D. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11:430–452, 1990.
19. W. Wasow. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, 6:78–78, 1952.
20. C. Xu and Y. Nie. Relaxed implementation of spectral methods for graph partitioning. In *Proc. of the 5th Int. Symp. on Solving Irregular Problems in Parallel (Irregular'98), August 1998, Berkeley, CA*, 1998.