Open access • Proceedings Article • DOI:10.1109/IPPS.1996.508075

Parallel multilevel graph partitioning — Source link

George Karypis, Vipin Kumar

Institutions: University of Minnesota

Published on: 15 Apr 1996 - International Conference on Parallel Processing

Topics: Graph partition, Parallel algorithm, Speedup, Computational complexity theory and Sparse matrix

A short version of this paper appears in Supercomputing 1995

The algorithms described in this paper are implemented by the
'METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System'.
METIS is available on WWW at URL: http://www.cs.umn.edu/˜metis

Analysis of Multilevel Graph Partitioning [*]

George Karypis and Vipin Kumar

University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
Technical Report: 95-037

{karypis, kumar}@cs.umn.edu

Last updated on March 27, 1998 at 5:56pm

Abstract

Recently, a number of researchers have investigated a class of algorithms that are based on multilevel graph partitioning that have moderate computational complexity, and provide excellent graph partitions. However, there exists little theoretical analysis that could explain the ability of multilevel algorithms to produce good partitions. In this paper we present such an analysis. We show under certain reasonable assumptions that even if no refinement is used in the uncoarsening phase, a good bisection of the coarser graph is worse than a good bisection of the finer graph by at most a small factor. We also show that the size of a good vertex-separator of the coarse graph projected to the finer graph (without performing refinement in the uncoarsening phase) is higher than the size of a good vertex-separator of the finer graph by at most a small factor.

Keywords: Multilevel Partitioning Methods, Fill Reducing Ordering, Numerical Linear Algebra.

1 Introduction

Graph partitioning is an important problem that has extensive applications in many areas, including scientific computing, VLSI design, and task scheduling. The problem is to partition the vertices of a graph in p roughly equal parts, such that the number of edges connecting vertices in different parts is minimized. For example, the solution of a sparse

[*]This work was supported by Army Research Office contract DA/DAAH04-95-1-0538 NSF grant CCR-9423082, and by Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute, Cray Research Inc, and by the Pittsburgh Supercomputing Center. Related papers are available via WWW at URL: **http://www.cs.umn.edu/˜karypis**

system of linear equations $Ax = b$ via iterative methods on a parallel computer gives rise to a graph partitioning problem. A key step in each iteration of these methods is the multiplication of a sparse matrix and a (dense) vector. The problem of minimizing communication in this step is identical to the problem of partitioning the graph corresponding to the matrix A [17]. If parallel direct methods are used to solve a sparse system of equations, then a graph partitioning algorithm can be used to compute a fill reducing ordering that lead to high degree of concurrency in the factorization phase [17, 5].

The graph partitioning problem is NP-complete. However, many algorithms have been developed that find reasonably good partitions. Spectral methods [28, 27, 13] have been shown to be quite effective for partitioning unstructured problems in a variety of applications, but have very high computational complexity. The MSB algorithm produces partitions that are as good as those produced by the original spectral bisection, but it is one to two orders of magnitude faster as it computes the Fiedler vector of the graph using a multilevel approach [28, 1]. Geometric partition methods [10, 11, 29, 22, 21, 23] are quite fast but they often provide worse partitions than those of more expensive methods such as spectral. Furthermore, geometric methods are applicable only if coordinate information for the graph is available.

Recently, a number of researches have investigated a class of algorithms that have moderate computational complexity, and provide excellent (even better than spectral) graph partitions [3, 13, 14]. The basic idea behind these algorithms is very simple. The graph G is first coarsened down to a few hundred vertices, a bisection of this much smaller graph is computed, and then this partition is projected back towards the original graph (finer graph) by periodically refining the partition. Since the finer graph has more degrees of freedom, such refinements usually decrease the edge-cut. This process, is graphically illustrated in Figure 1. These are called multilevel graph partitioning schemes. In particular, in [14] we have developed a multilevel graph partitioning scheme that produces high quality partitions that perform consistently better than the spectral methods, while requiring significantly less time (10 to 30 times less) than even multilevel spectral bisection. We also used our multilevel graph partitioning scheme to compute fill reducing orderings for sparse matrices [14]. Surprisingly, our scheme substantially outperforms the multiple minimum degree algorithm [19], which is the most commonly used method for computing fill reducing orderings of a sparse matrix.

From the experiments presented in Section 2.1 and those of other researchers [3, 13], it is clear that multilevel graph partitioning algorithms are able to find high quality partitions for a variety of unstructured graphs. However, there exists little theoretical analysis that could explain the ability of multilevel algorithms to produce good partitions. In this paper we present such an analysis. We show under certain reasonable assumptions that even if no refinement is used in the uncoarsening phase, a good bisection of the coarser graph is worse than a good bisection of the finer graph by at most a small factor. We also show that the size of a good vertex-separator of the coarse graph projected to the finer graph (without performing refinement in the uncoarsening phase) is higher than the size of a good vertex-separator of the finer graph by at most a small factor.

The rest of the paper is organized as follows. Section 2 briefly describes the multilevel graph bisection algorithm and summarizes experimental performance results from [14]. Section 3 analyzes the bisections and vertex-separators produced by the multilevel algorithm, and presents supporting experiments. Section 4 provides some concluding remarks.

2 Multilevel Graph Bisection

In this section we briefly describe the various phases of the multilevel algorithm. The reader should refer to [14] for further details.

Coarsening Phase During the coarsening phase, a sequence of smaller graphs $G_i = (V_i, E_i)$, is constructed from the original graph $G_0 = (V_0, E_0)$ such that $|V_i| > |V_{i+1}|$. Graph G_{i+1} is constructed from G_i by finding a maximal matching $M_i \subseteq E_i$ of G_i and collapsing together the vertices that are incident on each edge of the matching. In this process no more than two vertices are collapsed together because a matching of a graph is a set of edges, no two of which are incident on the same vertex. Vertices that are not incident on any edge of the matching, are simply copied over to G_{i+1}.

When vertices $v, u \in V_i$ are collapsed to form vertex $w \in V_{i+1}$, the weight of vertex w is set to be equal to the sum

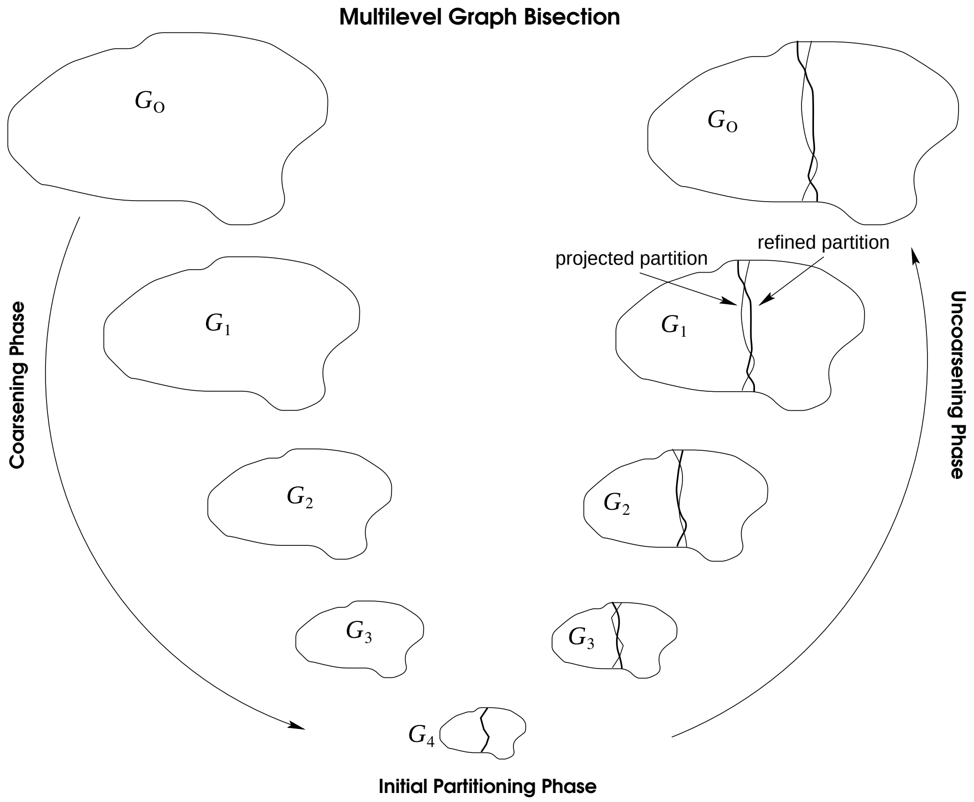

Figure 1: The various phases of the multilevel graph bisection. During the coarsening phase, the size of the graph is successively decreased; during the initial partitioning phase, a bisection of the smaller graph is computed; and during the uncoarsening phase, the bisection is successively refined as it is projected to the larger graphs. During the uncoarsening phase the light lines indicate projected partitions, and dark lines indicate partitions that were produced after refinement.

of the weights of vertices v and u, while the edges incident on w is set equal to the union of the edges incident on v and u minus the edge (v, u). If there is an edge that is incident to both on v and u, then the weight of this edge is set equal to the sum of the weights of these edges. Thus, during successive coarsening levels, the weight of both vertices and edges increases. A vertex in a graph G_i is call ***multinode*** if it contains more than one vertex of G_0.

Maximal matchings can be computed in different ways [14]. The method used to compute the matching greatly affects both the quality of the bisection, and the time required during the uncoarsening phase. Here we briefly describe two such matching schemes. The quality of the bisection for these two schemes is analyzed in Section 3.

The first scheme, which we called ***random matching*** (RM), computes the maximal matching by using a randomized algorithm [3, 13]. The RM scheme works as follows. The vertices of the graph are visited in random order. If a vertex u has not been matched yet, then an unmatched adjacent vertex v is randomly selected and the edge (u, v) is included in the matching. If there is no unmatched adjacent vertex v, then vertex u remains unmatched. The second scheme, which we call ***heavy-edge matching*** (HEM), computes a matching M_i, such that the weight of the edges in M_i is high. The HEM matching is computed using a randomized algorithm similar to the one used for RM. The vertices are again visited in random order. However, instead of randomly matching a vertex with one of its adjacent unmatched vertices, HEM matches it with the unmatched vertex that is connected with the heavier edge. As a result, the HEM scheme reduces the sum of the weights of the edges in the coarser graph by a larger amount than RM. In [14], we experimentally evaluated both the RM and HEM matching schemes, and we found that the HEM scheme produces consistently better results than RM, and the amount of time spent in refinement is less than that of RM.

Initial Partitioning Phase The second phase of a multilevel algorithm is to compute a balanced bisection of the coarsest graph $G_k = (V_k, E_k)$. In [14] we evaluated four different algorithms for partitioning the coarser graph. These are (a) spectral bisection (SB) [28, 1, 13], (b) Kernighan-Lin bisection (KL) [16, 4], (c) breadth-first region growing (GGP) [5, 7], and (d) greedy region growing (GGGP) [14]. We found that even though all four algorithms produce fairly similar initial partitions, the partitions produced by GGGP were consistently better.

Uncoarsening Phase During the uncoarsening phase, the partition of the coarsest graph G_k is projected back to the original graph by going through the graphs $G_{k-1}, G_{k-2}, \ldots, G_1$. Since each vertex $u \in V_i$ contains a distinct subset U of vertices of V_{i-1}, projecting the partition of G_i to G_{i-1} is done by simply assigning the vertices in U to the same part that vertex u belongs to.

Furthermore, even if the partition of G_i is at a local minima, the partition of G_{i-1} obtained by this projection may not be at a local minima. Since G_{i-1} is finer, it has more degrees of freedom that can be used to further improve the partition and thus decrease the edge-cut. Hence, it may still be possible to improve the the partition of G_{i-1} obtained by the projection by using local refinement heuristics. For this reason, after projecting a partition, a partition refinement algorithm is used.

The basic purpose of a partition refinement algorithm is to select two subsets of vertices, one from each part such that when swapped the resulting partition has smaller edge-cut. Specifically, if A and B are the two parts of the bisection, a refinement algorithm selects $A' \subset A$ and $B' \subset B$ such that $A \backslash A' \cup B'$ and $B \backslash B' \cup A'$ is a bisection with a smaller edge-cut. The sets A' and B' are usually constructed incrementally, and a number of algorithms have been proposed for their construction.

A class of algorithms that tend to produce very good results are those that are based on the Kernighan-Lin partition algorithm [16, 4, 13]. These algorithms associate with each vertex v a quantity called *gain* which is the decrease (or increase) in the edge-cut if v is moved to the other part. These algorithms proceed by repeatedly selecting vertices with the highest gains from each part, inserting them into A' and B', and updating the gains of the remaining vertices. In [14] we evaluated four different refinement algorithms that belong to this class and we found that some of them are both effective in reducing the edge-cut and also require very little time.

Matrix Name	No. of Vertices	No. of Edges	Description
3ELT	4720	13722	2D Finite element mesh
4ELT	15606	45878	2D Finite element mesh
BCSSTK30	28294	1007284	3D Stiffness matrix
BCSSTK32	44609	985046	3D Stiffness matrix
BRACK2	62631	366559	3D Finite element mesh
CANT	54195	1960797	3D Stiffness matrix
COPTER2	55476	352238	3D Finite element mesh
CYLINDER93	45594	1786726	3D Stiffness matrix
FINAN512	74752	261120	Linear programming
INPRO1	46949	1117809	3D Stiffness matrix
LHR71	70304	1449248	3D Coefficient matrix
MAP1	267241	334931	Highway network
ROTOR	99617	662431	3D Finite element mesh
S38584.1	22143	35608	Sequential circuit
SHELL93	181200	2313765	3D Stiffness matrix
SHYY161	76480	152002	CFD/Navier-Stokes
TROLL	213453	5885829	3D Stiffness matrix
WAVE	156317	1059331	3D Finite element mesh
WHITAKER3	9800	28989	2D Finite element mesh

Table 1: Various matrices used in evaluating the multilevel graph partitioning and sparse matrix ordering algorithm.

2.1 Comparison with Other Partitioning and Ordering Schemes

Despite the simplicity of the multilevel partitioning algorithm, it has been found to produce high quality partitions that are equally good or better than those produced by more sophisticated algorithms that require much more time [28, 1]. In [14] we presented an extensive comparison between our multilevel algorithm that uses the HEM matching scheme and other widely used schemes. In this section we briefly summarize these results for the matrices in Table 1.

Figure 2 shows the relative performance of our multilevel algorithm compared to multilevel spectral bisection algorithm (MSB) [1], implemented in the Chaco [12] graph partitioning package. For each matrix we plot the ratio of the edge-cut of our multilevel algorithm to the edge-cut of the MSB algorithm. Ratios that are less than one indicate that our multilevel algorithm produces better partitions than MSB. From this figure we can see that for almost all the problems, our algorithm produces partitions that have smaller edge-cuts than those produced by MSB. In some cases, the improvement is as high as 70%. Furthermore, the time required by our multilevel algorithm is significantly smaller than that required by MSB. Our algorithm is usually 10 times faster for small problems, and 15 to 35 times faster for larger problems [14].

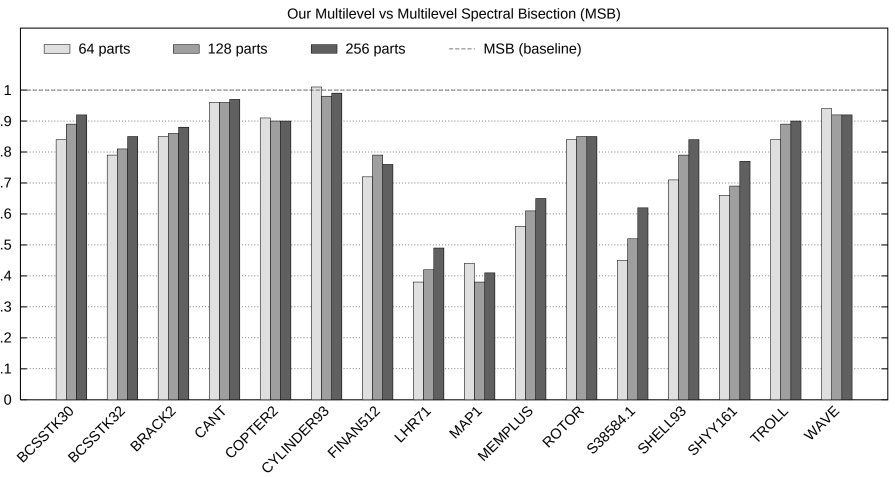

Figure 2: Quality of our multilevel algorithm compared to the multilevel spectral bisection algorithm. For each matrix, the ratio of the cut-size of our multilevel algorithm to that of the MSB algorithm is plotted for 64-, 128- and 256-way partitions. Bars under the baseline indicate that the multilevel algorithm performs better.

The multilevel graph partitioning algorithm can be used to find a fill reducing ordering for a symmetric sparse matrix via recursive nested dissection. Figure 3 shows the quality of our multilevel nested dissection ordering algorithm (MLND) compared to the multiple minimum degree (MMD) [6, 19], and spectral nested dissection (SND) [26]. These graphs were produced by dividing the number of operations required to factor a matrix using MLND and SND to the number of operations required by MMD. For both MLND and SND, a vertex separator is computed from an edge separator by finding the minimum vertex cover [24, 25]. MMD has been found to produce very good orderings and is widely used to order sparse matrices for factorization on serial computers, while SND is used to order sparse matrices for factorization on parallel computers because it produces orderings with balanced elimination trees [15, 9].

From this figure we see that compared against MMD, our algorithm produces better orderings for 10 out of the 13 matrices, and compared against SND, it produces better ordering for all 13 matrices. Also, from Figure 3 we see that MLND does consistently better than MMD as the size of the matrices increases and as the matrices become more unstructured. When all 13 test matrices are considered, MMD produces orderings that require a total of 711 billion operations, whereas the orderings produced by MLND require only 274 billion operations. Thus, the ensemble of 13 matrices can be factored roughly 2.6 times faster (even on a serial computer) if ordered with MLND.

However, another even more important advantage of MLND over MMD, is that it produces orderings that exhibit significantly more concurrency than MMD. The elimination trees produced by MMD exhibit little concurrency (long and slender), and are unbalanced so that subtree-to-subcube mappings lead to significant load imbalances [17, 5, 8]. One the other hand, orderings based on nested dissection produce elimination trees that have both more concurrency and better balance [15, 9]. Therefore, when the factorization is performed in parallel, the better utilization of the processors can cause the ratio of the run time of parallel factorization algorithms using MMD and MLND to be substantially higher than the ratio of their respective operation counts.

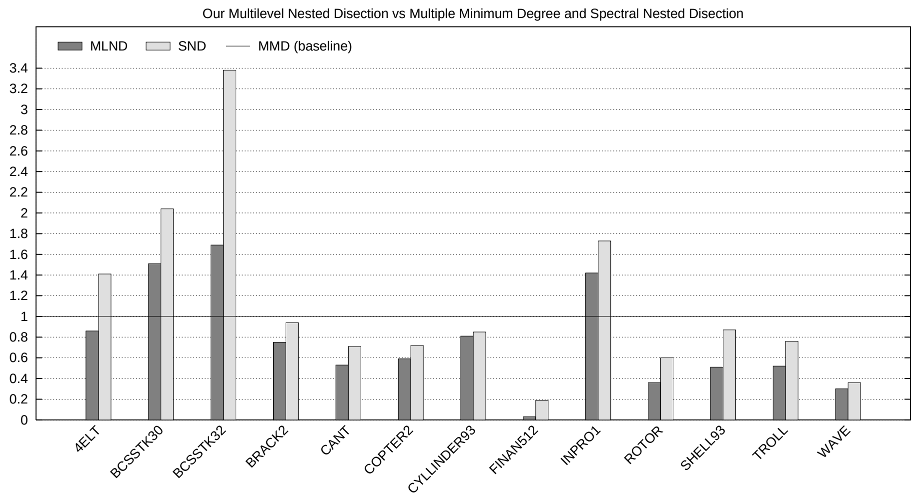

Figure 3: Quality of our multilevel nested dissection relative to the multiple minimum degree, and the spectral nested dissection algorithm. Bars under the baseline indicate that the MLND algorithm performs better.

3 Analysis

The central theme of the analysis is as follows. If the graph is coarsen "perfectly", then the coarsest graph will be an exact replica of the finer (original) graph except that it will be smaller. Hence, an optimal bisection of this coarsest (smaller) graph will also be an optimal bisection of the finer graph. If the graph has been coarsened (using a perfect coarsening scheme) enough times, then it becomes small enough that we can find a "near-optimal" bisection in a reasonable amount of time. This near-optimal bisection of the coarsest graph will also be a good bisection of the original graph. If the coarsening scheme is not good, then it is entirely possible that a near-optimal bisection of the coarsest graph is an arbitrarily bad bisection of the finer (original) graph. The matching strategies (RM and HEM) used in our multilevel algorithm do not lead to optimal coarsening. Hence, a near-optimal bisection of the coarsest graph obtained using RM or HEM, is not guaranteed to be a near-optimal bisection of the finer graph. The analysis in Section 3.2 shows that under some reasonable assumptions, the edge-cut of a near-optimal bisection of the coarser graph is larger than the edge-cut of a near-optimal bisection of the finer graph only by a small constant factor. The analysis also show that this "penalty" factor is even smaller when the HEM scheme is used (instead of RM).

Similarly, if the graph is coarsened perfectly, then a near-optimal vertex separator of the coarser graph can be projected to the finer graph to construct a near-optimal separator of the finer graph [1]. As for the case of the edge-cut, if the coarsening scheme is not good, then the projected separator of the coarser graph can lead to an arbitrarily bad separator of the finer graph. The vertex-separator analysis in Section 3.3 shows that under reasonable assumptions, the projection of a near-optimal separator of the coarser graph leads to a separator for the finer graph that is worse than a near-optimal separator (for the finer graph) only by a small constant factor.

Both of these analyses show that even if no refinement is performed during the uncoarsening phase, the bisection of the coarsest graph is also a good bisection of the original graph, especially if HEM is used for coarsening. These observations are supported by experimental results in both cases.

3.1 Definitions and Assumptions

Analyzing the quality of the bisections produced by multilevel graph partitioning algorithms is particularly hard, and can only be done if certain assumptions are made about the original graphs and the coarsening process. In the rest of this section we present these assumptions and some notation that will be used throughout the analysis.

[1] A straightforward projection of the separator to the finer graph will simply increase the number of vertices in the separator. But as discussed in Section 3.3, unnecessary vertices from the projected separator can be dropped to construct a good separator. It can be shown that if the separator of the coarsest graph is optimal and the coarsest graph was constructed by perfect matching, after dropping these "extra" vertices, the resulting separator will be an optimal separator of the finer graph,

Let $G_0 = (V_0, E_0)$ be the original graph, and $G_i = (V_i, E_i)$ be the ith level coarser graph. For each graph G_i, let $\underline{W(M_i)}$ be the sum of the weights of the edges in the matching used to obtain G_{i+1}, $\underline{W(E_i)}$ be the sum of the weights of the edges, $\underline{\omega_i}$ be the average edge-weight, $\underline{d_i}$ be the average degree of a vertex, and $\underline{C_i}$ be the size of the edge-cut (*i.e.*, the weight of the edges crossing parts).

To simplify the presentation of the analysis we assume that at each successive level of coarsening, the number of vertices reduce by a factor of 2, *i.e.*, $\underline{|V_i| = 2|V_{i+1}|}$. Consequently, the matching at level i contains $|V_i|/2$ edges; hence, the weight of the matching is equal to $\underline{W(M_i) = \delta\omega_i|V_i|/2}$, where δ is a constant that captures the properties of RM and HEM. In particular, $\delta = 1$ for RM (because the matched edges in RM are chosen randomly), and $\delta \geq 1$ for HEM (because the HEM prefers edges with higher weight). Also, to simplify the analysis we assume that the average degree of the successive coarser graphs changes at a constant rate, *i.e.*, $\underline{d_i/d_{i+1} = \beta}$. Note that if $\beta < 1$, the average degree increases, if $\beta > 1$ the average degree decreases, and if $\beta = 1$ the average degree remains the same.

In the rest of the analysis we assume that the following are true.

Assumption 1 (Small Separators) *There are constants α and γ such that each graph G_i has a balanced separator of size less than or equal to $\underline{\alpha|V_i|^\gamma}$.*

Assumption 2 (Size of the Edge-Cut) *The edge-cut of G_i is proportional to $\underline{\alpha d_i\omega_i|V_i|^\gamma}$. That is, C_i is proportional to the size of the balanced separator, the average degree of the graph, and the average weight of each edge.*

In the case of graphs arising in finite element applications, the small separator assumption is true. In particular, for planar graphs $\gamma = 0.5$ [18], and for the graphs that correspond to 3D finite element meshes, $\gamma = 2/3$ [22]. Assumption 2 follows directly from Assumption 1 and corresponds to the bisection in which the separator is the boundary of one of the two parts. For graphs arising in finite element applications, the edge-cut cannot be asymptotically smaller because vertices have bounded degrees, and this will lead to an asymptotically smaller vertex separator.

3.2 Edge-Cut Analysis

The multilevel algorithm can be viewed as an approximation algorithm since every successively coarse graph becomes a successively rougher approximation of the original graph. Since more and more features of the original graph are eliminated in this process, the coarser graphs have bisections with larger edge-cut than the original graph. It is natural to ask how bad can the edge-cut of the coarser graph get with respect to the edge-cut of the original graph. In the rest of this section we establish a relation between C_i and C_{i-1} and we use it to express C_i as a function of C_0 for graphs arising in 2D and 3D finite element applications.

From Assumption 2 we know that the edge-cut of G_i depends on the average degree of its vertices (d_i), and on the average weight of its edges (ω_i). From the assumptions in Section 3.1 we know how d_i is related to d_{i-1}; thus, in order to express C_i as a function of C_{i-1} we need to compute how ω_i and ω_{i-1} are related. From the definition of ω_i we have

$$\omega_i = \frac{W(E_i)}{|V_i|(d_i/2)}.$$

Recall that G_i is obtained from G_{i-1} by collapsing the vertices incident on the edges of the matching M_{i-1}. Thus, by moving from G_{i-1} to G_i, the sum of the weights of the edges is decreased by $W(M_{i-1})$. Hence, $W(E_i)$ is

$$W(E_i) \;=\; W(E_{i-1}) - W(M_{i-1}) \;=\; d_{i-1}\omega_{i-1}\frac{|V_{i-1}|}{2} - \delta\omega_{i-1}\frac{|V_{i-1}|}{2} \;=\; (d_{i-1} - \delta)\omega_{i-1}\frac{|V_{i-1}|}{2}. \tag{1}$$

By substituting this equation into the definition of ω_i we have

$$\omega_i \;=\; \frac{(d_{i-1} - \delta)|V_{i-1}|\omega_{i-1}}{d_i|V_i|} \;=\; 2\frac{d_{i-1} - \delta}{d_i}\omega_{i-1} \;=\; 2\beta\left(1 - \frac{\delta\beta^{i-1}}{d_0}\right)\omega_{i-1}. \tag{2}$$

From this equation and Assumption 2 we can prove the following theorem that express C_i as a function of C_{i-1}.

Theorem 1 *The size of the edge-cut for graph G_i for $i \geq 1$ is*

$$C_i \propto 2^{1-\gamma}\left(1 - \frac{\delta\beta^{i-1}}{d_0}\right)C_{i-1}. \tag{3}$$

Proof.

$$C_i \propto \alpha d_i \omega_i |V_i|^\gamma \;=\; \alpha\frac{d_{i-1}}{\beta}2\beta\left(1 - \frac{\delta\beta^{i-1}}{d_0}\right)\omega_{i-1}|V_i|^\gamma \;=\; 2\left(1 - \frac{\delta\beta^{i-1}}{d_0}\right)(\alpha d_{i-1}\omega_{i-1}|V_{i-1}|^\gamma)2^{-\gamma}$$

Since $C_{i-1} \propto \alpha d_{i-1}\omega_{i-1}|V_{i-1}|^\gamma$ we have that

$$C_i \propto 2^{1-\gamma}\left(1 - \frac{\delta\beta^{i-1}}{d_0}\right)C_{i-1}.$$

■

Equation 3 reveals significant information about the quality of edge-cut when it is computed at a coarse graph. In particular, for a given graph, the increase in the edge-cut between successive coarse graphs decreases as either δ or β increases. That is, if the weight of the edges in the independent set used to coarsen the graph is much higher than the average weight of the edges ($\delta > 1$), then the penalty paid for finding a partition at the coarse graph is smaller. Similarly, as the average degree of the coarse graph decreases ($\beta > 1$), again the increase of the edge-cut at the coarse graph is smaller. In the next two sections we see how Equation 3 applies to graphs that correspond to 2D and 3D finite element meshes with triangular and tetrahedron elements, respectively.

3.2.1 2D Finite Element Meshes

The 2D finite elements meshes correspond to planar graphs. Furthermore, when the elements of the mesh are triangles (*e.g.*, when the finite element mesh is generated using Delaunay triangulation), then the graph that corresponds to the interior of the mesh is maximally planar. For the rest of this section we use this correspondence and we only concentrate on maximally planar graphs.

Planar graphs have been extensively studied and a great deal of properties are known about them. In particular, for maximally planar graphs $\gamma = 0.5$ [18], and $d_0 \approx 6$. Also, in [18] it was shown that edge contraction also preserves maximal planarity; thus, $\beta = 1$.

From Equation 3, and for RM (*i.e.*, $\delta = 1$), we have that

$$C_i^{2D-RM} \;\propto\; 1.18C_{i-1}^{2D-RM} \;=\; 1.18^i C_0 \tag{4}$$

Thus, the edge-cut increases only by 18% at each successive coarsening level. For instance, the edge-cut after 10 coarsening levels is only 5.2 times worse than the edge-cut of the original graph. However, the size of G_{10} is smaller than G_0 by a factor of 1024, so it is much quicker to find a good partition of G_{10} than of G_0.

As discussed in Section 3.2, the increase in the edge-cut at successive coarsening levels is smaller when HEM is used, because in this case $\delta > 1$. For instance if $\delta = 1.3$ (as observed in our experiments), then $C_i^{2D-HEM} \propto 1.11C_{i-1}^{2D-HEM}$. In this case, after 10 coarsening levels, the edge-cut is only 2.8 times worse than the edge-cut of G_0.

3.2.2 3D Finite Element Meshes

The graphs that correspond to 3D finite element meshes do not correspond to any extensively studied class of graphs as the 2D finite element graphs did. Nevertheless, for these type of graphs it is known that $\gamma = 2/3$ [22] and that for most finite element applications d_0 ranges between 12 and 16 [2]. However, in order to apply Equation 3, we need to know the value of β. Unlike maximally planar graphs, for which the average degree of successive coarser graphs remains the same, the average degree of 3D finite element graphs does not always remain the same. In particular, if

$d_0 > 12$, and RM is used to coarsen the graph, d_i increases in successive coarser graphs. However, if HEM is used to coarsen the graph, then the average degree of the graph actually decreases in successive coarsening levels as discussed later.

Theorem 2 (Average Degree of 3D Graphs) *The average degree of the interior vertices of the ith level coarser graph G_i, of a graph G_0 that corresponds to a 3D finite element mesh with tetrahedron elements is*

$$d_i \;=\; 2\left(d_{i-1} - 7 + \frac{12}{d_{i-1}}\right). \tag{5}$$

Proof. Consider a 3D finite element mesh with tetrahedron elements. Let, Φ_i be the number of tetrahedrons in graph G_i. The number of interior vertices, edges, and tetrahedrons are related by Euler's formula [2] as follows

$$|E_i| = |\Phi_i| + |V_i|. \tag{6}$$

Let ζ_i be the average number of triangular phases incident on an edge of graph G_i. Since, each tetrahedron has 6 edges, ζ_i is equal to

$$\zeta_i \;=\; \frac{6|\Phi_i|}{|E_i|} \;=\; \frac{6(|E_i| - |V_i|)}{|E_i|} \;=\; 6 - \frac{12}{d_i}. \tag{7}$$

During coarsening, for each edge that gets collapsed, ζ_i triangles are eliminated. Consequently, ζ_i edges are eliminated, in addition to the one that gets collapsed. Therefore, the number of edges at the next level coarser graph is

$$|E_i| \;=\; |E_{i-1}| - (\zeta_{i-1} + 1)\frac{|V_{i-1}|}{2} \;=\; |E_{i-1}| - \left(7 - \frac{12}{d_{i-1}}\right)\frac{|E_{i-1}|}{d_{i-1}} \;=\; \left(1 - \frac{7}{d_{i-1}} + \frac{12}{d_{i-1}^2}\right)|E_{i-1}|. \tag{8}$$

Therefore, the degree of the ith level coarse graph is

$$d_i \;=\; \frac{2|E_i|}{|V_i|} \;=\; \frac{2(1 - 7/d_{i-1} + 12/d_{i-1}^2)|E_i|}{|V_{i-1}|/2} \;=\; 2\left(1 - \frac{7}{d_{i-1}} + \frac{12}{d_{i-1}^2}\right)d_{i-1} \;=\; 2\left(d_{i-1} - 7 + \frac{12}{d_{i-1}}\right).$$

■

From Equation 5 we have that when $d_0 = 12$, $d_i = 12$ for all coarse graphs. However, if $d_0 < 12$, d_i decreases whereas if $d_0 > 12$, d_i increases at each successive coarse graph.

Thus, when $d_0 = 12$, $\beta = 1$, and from Equation 3 we have that the edge-cut for 3D finite element graphs when RM is used to coarsen the graph (*i.e.*, $\delta = 1$) is

$$C_i^{3D-RM} \;\propto\; 1.15 C_{i-1}^{3D-RM} = 1.15^i C_0^{3D-RM}. \tag{9}$$

Comparing this equation with Equation 4, we see that the increase in the edge-cut at successive coarsening levels is smaller for 3D graphs than it is for 2D graphs. As it was the case with 2D graphs, the increase in the edge-cut is small, compared to the reduction in the graph size. If HEM is used to coarsen the graph, then $\delta > 1$, and the decrease in quality is even smaller.

The increase in the average degree of the graph can be controlled if coarsening is done in a certain way. In deriving Equation 8, we assumed that each edge in the matching is such that the number of triangular phases incident on it is equal to the average (ζ_i). This assumption is valid when RM is used to coarsen the graph which randomly selects unmatched edges. However, a different matching can be used that selects those edges that have a large number of triangular phases incident on it. In this case, the number of edges will be smaller than that indicated by Equation 8, leading to a smaller degree d_i.

We claim that by selecting edges with high weight, HEM tends to select edges that have a large number of triangular phases incident on them. To see this, consider the first level coarser graph G_1. Let e_h, and e_l be two edges of G_1, such that e_h has higher weight than e_l. Let E^h and E^l be the subsets of edges from the original graph G_0 that have been merged to form e_h and e_l respectively. Note that $|E^h| > |E^l|$, since e_h has larger weight that e_l. Due to the nature of the coarsening, the number of triangular phases incident on e_h (e_l) is the union of the triangular phases incident on the edges of E^h (E_l) that have not been collapsed during the coarsening. Since $|E^h| > |E^l|$, the number of triangular phases incident on e_h is larger than those incident on e_l. Thus, HEM creates coarser graphs that have smaller average degree than those created by RM. In fact, our experiments show (Figure 5) that for HEM, the average degree of the coarse graphs not only does not increase but it actually decreases.

3.2.3 Experimental Results

To verify the analysis presented in Sections 3.2.1 and 3.2.2, we instrumented our multilevel algorithm to report various statistics during coarsening. In the rest of this section we present results for the following four matrices: 4ELT is a 2D finite element mesh; BRACK2 and WAVE are 3D finite element meshes; and CANT is the graph that corresponds to a 3D finite element mesh with multiple degrees of freedom per mesh point. This set of matrices is a representative sample of the matrices shown in Table 1.

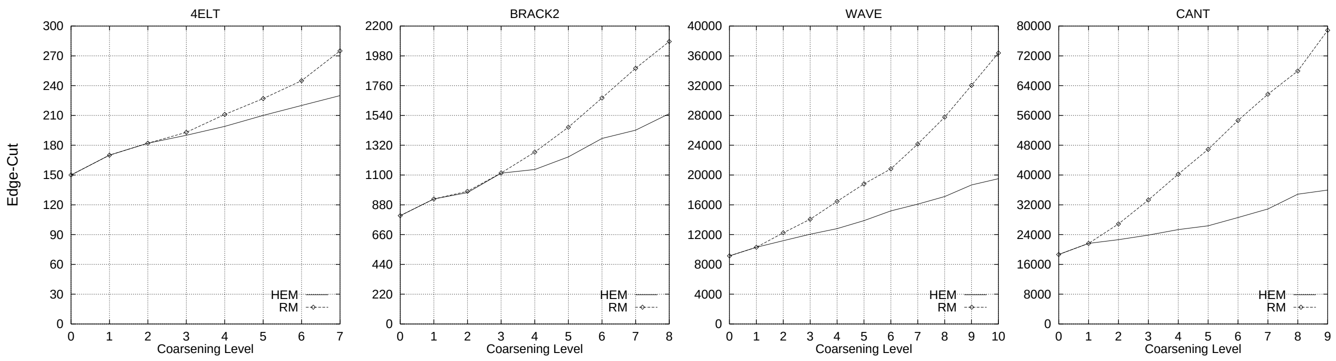

Figure 4: The increase in the edge-cut at successive coarsening levels.

Figure 4 shows the edge-cut C_i for successive coarsening levels for both the RM and HEM coarsening schemes. The edge-cut at the ith level was obtained by performing i coarsening levels to obtain G_i, and then using our multilevel algorithm to find a good partition of this coarser graph [2]. The plotted value of C_i is the minimum of the edge-cuts produced by the multilevel algorithm and SB. From this graph we see that as indicated by our analysis, for all four matrices, the edge-cut C_i increases slowly at successive coarsening levels. When RM is used to coarsen the graphs, the edge-cut at the last coarsening level is only 1.8, 2.4, 2.9, and 4.2 times worse than C_0 for 4ELT, BRACK2, WAVE, and CANT respectively. This increase in the edge-cut is actually lower than the one predicted by the analysis. This should not be surprising since Equation 3 is only an upper bound. Also, from Figure 4 we see that when HEM is used to coarsen the graph, the edge-cuts at the coarser graphs and their rate of increase at successive coarsening levels are smaller than those of the RM coarsening scheme. This is also predicted by our analysis since for HEM, $\delta > 1$. Furthermore, for 3D finite element meshes, HEM tends to decrease the average degree of the graph with successive coarsening levels as shown in Figure 5.

Figure 5 shows the average degree of the graphs at successive coarsening levels for both RM and HEM. For 4ELT, we see that for both coarsening schemes, the average degrees are similar and they slowly decrease. This decrease is due to the boundary elements of the mesh. However, for the 3D finite element meshes (BRACK2 and WAVE), the average degree of the graphs when RM is used to coarsen them, increases at successive coarsening levels. This is consistent with the analysis presented in Section 3.2.2, which indicated that the average degree of 3D finite element meshes increases when RM is used. Also, comparing the degrees of BRACK2 and WAVE, we see that the degrees increase at a higher rate if d_0 is high. This again follows directly from Equation 5. However, the increase in the average

[2]We used the multilevel graph bisection to compute effectively a new-optimal separator for the graph, since it has the best performance and reasonable cost. We also used the spectral bisection (SB) on each coarse graph to check if our multilevel algorithm was producing better partitions. In most cases, the multilevel algorithm produced better partitions than the spectral algorithm.

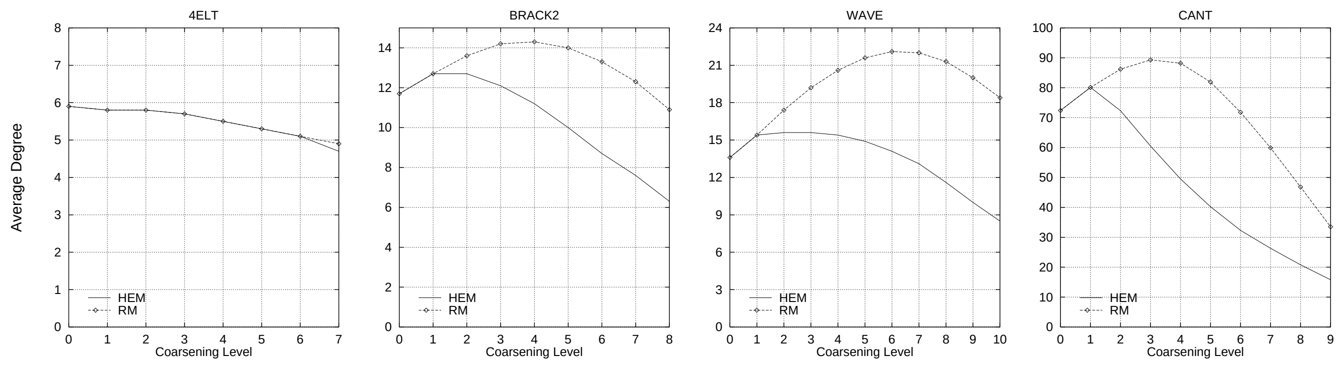

Figure 5: The average degrees at successive coarsening levels.

degree of the graph tappers off after a number of coarsening levels because of the boundary elements (like it was for the 2D case).

Our analysis in Section 3.2.2 also predicted that if HEM is used to coarsen the graph, then the average degree of 3D finite element meshes will increase at a slower rate than RM. This is clearly illustrated in Figure 5. The average degree of the graph increases for the 1st level coarser graph (d_1) because HEM and RM are equivalent for the first coarsening level (initially all edges have weight of one). The average degree of subsequent coarser graphs, not only do not increase but actually they decrease quite substantially. The advantages of the decreasing average degree can be seen by comparing the edge-cut of RM and HEM at the coarsest graph for BRACK2, WAVE, and CANT. For instance, at the last coarsening level, the HEM edge-cut for WAVE is only 1.8 times higher while the RM edge-cut is 2.9 times higher.

Since CANT is the graph of the coefficient matrix of a 3D mesh with multiple degrees of freedom per mesh-point, it does not correspond to the graphs analyzed in Section 3.2.2. Consequently, Equations 9 and 5 do not apply for this type of graphs. However, Equation 3 still applies, but determining the average degree of successive coarser graphs is particularly difficult since Euler's formula is not valid in this case. However, from Figures 4 and 5 we can see how the edge-cut and the average degree of the graph changes at successive coarsening levels. From Figure 5 we can see that for RM, d_i increases during the first five coarsening levels, and then decreases, whereas for HEM, d_i decreases rapidly after the first coarsening level. The advantage of HEM over RM can be easily seen in Figure 4 for which the HEM edge-cut is only 1.9 times higher while the RM edge-cut is 4.2 times higher. As it was the case for 3D meshes (Section 3.2.2), the average degree of successive coarser graphs decreases because HEM selects edges that have a large number of triangular phases incident on them while RM does not.

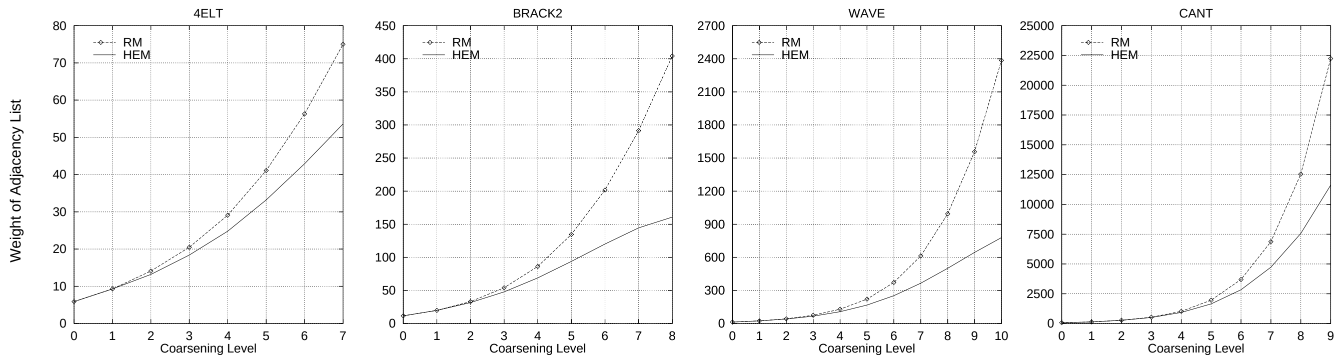

Figure 6: The average weight of the adjacency lists at successive coarsening levels. The weight of the adjacency list at level i is $d_i\omega_i$.

Finally, Figure 6 shows the average weight of the adjacency list of a vertex for successive coarsening levels. Recall that the average weight of the adjacency list at the ith level is equal to $d_i\omega_i$ and from Assumption 2 it is directly related to the edge-cut of the graph. From Figure 6 we see that for all four matrices, the weight of the adjacency lists when HEM is used to coarsen the graph is smaller than that for RM.

3.3 Vertex Separator Analysis

In the previous section we showed that the edge-cut of the coarser graph is higher than the edge-cut of the original graph G_0 by a small factor. Consequently, from Assumption 2, this implies that the vertex separator induced by the bisection of the coarser graph is larger than the separator of the original graph by a small factor. In this section we analyze the coarsening process by looking at the vertex separators. In particular, we will show that the vertex separator obtained in the coarse graph is also a small vertex separator in the original graph. This analysis is focused on maximal planar graphs that satisfy the assumptions in Section 3. Furthermore, it is assumed that the separator forms a simple path or cycle [20]. This analysis can be extended to the graphs that correspond to 3D finite element meshes, when the separators are simple surfaces.

Consider the kth level coarse graph $G_k = (V_k, E_k)$. From the small separator assumption, we know that G_k has a separator S_k that contains no more than $\alpha\sqrt{|V_k|}$ vertices (recall that $\gamma = 0.5$ for planar graphs [18]). Let S'_0 be the union of the vertices of S_k projected to G_0. S'_0 forms a balanced separator of G_0 and its size is

$$|S'_0| = 2^k|S_k| \le \alpha 2^k\sqrt{|V_k|} = \alpha 2^k\sqrt{|V_0|/2^k} = \alpha(\sqrt{2})^k\sqrt{|V_0|}.$$

However, applying the small separator assumption to the original graph, the size of the separator of G_0 is $|S_0| \le \alpha\sqrt{|V_0|}$. In the limiting case, when $k = O(\log|V_0|)$, we have that $(\sqrt{2})^k = \sqrt{|V_0|}$, in which case $|S'_0| \le O(|V_0|)$. Thus, S'_0 contains $O(\sqrt{|V_0|})$ more vertices than a small separator. However, not all the vertices in S'_0 are required to form a separator for G_0, and a smaller separator can be constructed from S'_0 by *dropping* some vertices.

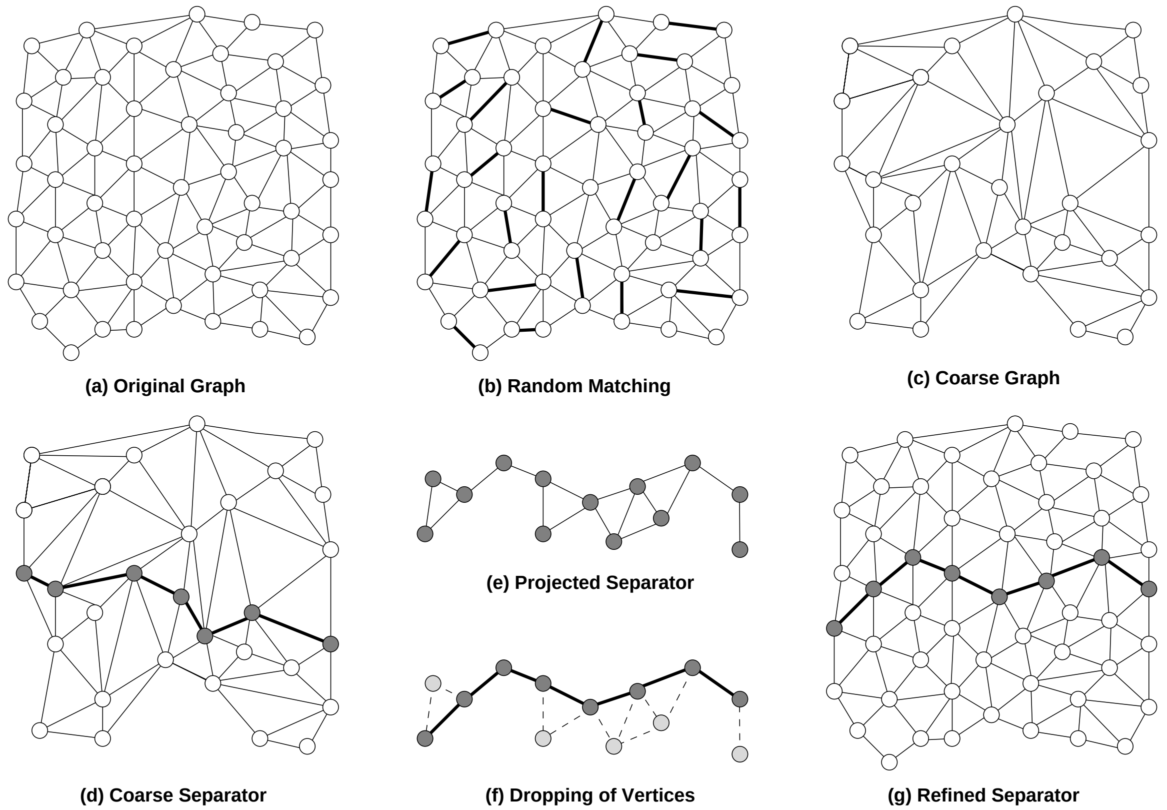

Figure 7: The sequence of one level coarsening, finding a separator for the coarse graph, projecting the separator to the original graph, and refining the separator by dropping vertices.

Figure 7 illustrates the basic idea behind the processes of vertex dropping. In this figure, the 1st level coarser graph is constructed using RM, and a separator of this coarse graph is computed (Figure 7(d)). Figure 7(e) shows the projected separator S_0' that corresponds to S_1. Note that not all the vertices of S_0' are necessary to form a separator for G_0. As Figure 7(f) illustrates certain vertices can be dropped. In the rest of this section we compute the average number of vertices being dropped in successive uncoarsening levels.

Consider the graph G_1, and let P_1 be a simple path or cycle of G_1. Let F be the subgraph of G_0 induced by the vertices of P_1 projected onto graph G_0. The subgraph F contains $2|P_1|$ vertices, and on the average these vertices do not form a simple path or cycle. In fact, the following lemma holds.

Lemma 1 (Path Projection) *Let G_1 be a one level coarse graph obtained from G_0 using random perfect coarsening, and let P_1 be a simple path of G_1, between vertices u_1 and v_1. Let F be the subgraph of G_0 that has P_1 as its minor, and let u_0 and v_0 be the vertices of F that are mapped onto u_1 and v_1 respectively. On the average, the shortest path between u_0 and v_0 in F contains less than $1.5|P_1|$ vertices.*

Proof. Let E_{P_1} be the edges of P_1, E_F be the edges of F, and $m = |P_1|$. Since P_1 is a simple path it contains $m - 1$ edges; thus $W(E_{P_1}) = (m - 1)\omega_1$. From Equation 2 in the case of maximally planar graphs, we have that $\omega_1 = 5/3$; thus, $W(E_{P_1}) = 5(m - 1)/3$.

The set E_F contains (a) all the edges in E_{P_1} and (b) the edges of the matching used to coarsen the graph, that match vertices of F. Thus, $|E_F| = W(E_F) = W(E_{P_1}) + m = 8m/3 - 5/3 \approx 8m/3$. If the vertices of F formed a simple path between u_0 and v_0, then $|E_F|$ should be equal to $2m - 1$, however E_F contains an extra $2m/3$ edges. Figure 8(a)-(b) illustrates this process. In Figure 8(a) the path P_1 between u_1 and v_1 is shown. The subgraph F of G_0 that has P_1 as its minor is shown in Figure 8(b).

Assume there is a simple path P in F between u_0 and v_0 that traverses all the vertices of F. P contains $2m$ vertices and $2m - 1$ edges. Thus, there is an additional $2m/3$ edges in E_F not included in P that can be used to decrease the length of the path between u_0 and v_0 (Figure 8(c)). We will refer to these $2m/3$ edges as ***jump-edges***.

Consider the shortest path from u_0 to v_0 in F. This path, upon entering a vertex w_i, it will go to vertex w_{i+2} if there is a jump-edge starting at w_i. In this case, a single edge traversal covers two vertices of P, namely w_i and w_{i+1}. If there is no jump-edge starting at w_i, the shortest path will go to vertex w_{i+1}, in which case it covers only one vertex of P, namely w_i. The probability p that a vertex is the source of a jump-edge is

$$p = \frac{2m/3}{2m - 2} \approx \frac{1}{3},$$

since each one of the $2m/3$ jump-edges can start from any vertex excluding the two last vertices. Thus, a shortest path from u_0 to v_0 that enters vertex w_i, goes to vertex w_{i+1} with probability $1 - p$, and to vertex w_{i+2} with probability p. The expected number of vertices that will be covered in a single step is $(1 - p) + 2p = 4/3$. The total number of steps k required to cover all $2m$ vertices in P is

$$\frac{4}{3}k = 2m \Rightarrow k = \frac{3}{2}m = 1.5m.$$

Therefore, the shortest path from u_0 to v_0 contains $1.5m$ vertices on the average.

In the case that there is no simple path in F between vertices u_0 and v_0 that traverses all the vertices, then the length of the shortest path between u_0 and v_0 will actually be smaller than $1.5m$.

■

The path projection lemma is very powerful and can be used to compute the size of the projected separator as a function of the separator of the coarser graph. Furthermore, as the next lemma shows, it can also be used to show that the sub-optimality of a separator at a coarser graph, decreases as this separator is projected to successively finer graphs.

Lemma 2 (Simple Separator) *Let G_k be the k level coarse graph obtained from G_0 using random matching. There is a separator for G_0 whose size is bounded by $\phi 0.75^k|V_0|$ for some constant ϕ.*

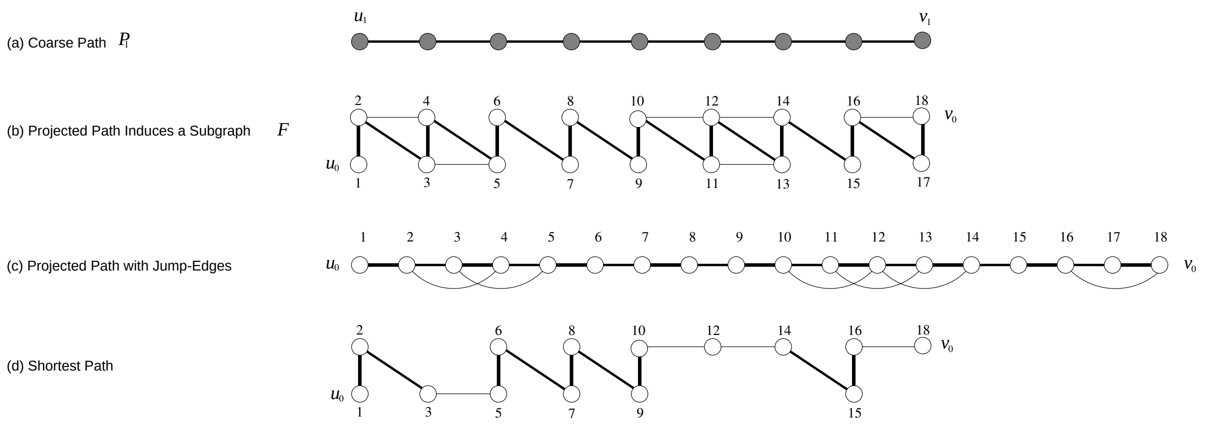

Figure 8: Outline of the proof of the path projection lemma. (a) A simple path P_1 between vertices u_1 and v_1 in the coarse graph. (b) The graph F induced in the next level finer graph by expanding the vertices and edges in P_1. Also, the dark edges show a simple path between u_0 and v_1 that traverses all the vertices, while the light edges are other edges of F not in this path. (c) A redrawing of F that clearly shows the path between u_0 and v_0 and the edges not belonging in the path (jump-edges). (d) The shortest path in F from u_0 to v_0.

Proof. Let S_k be a balanced separator that forms a simple path or cycle in G_k. There is a constant $\phi < 1$ such that, $|S_k| \le \phi|V_k| = \phi|V_0|/2^k$. Since, G_k was obtained from G_{k-1} using random matching, Lemma 1 applies, yielding a separator S_{k-1} of size $\phi 1.5|V_k|$. However, since RM is used to coarsen the graph, the path projection lemma also holds for S_{k-1} when projected to G_{k-2}. Thus, by k applications of Lemma 1 we have that the size of the projected separator S_0 of the original graph is

$$|S_0| \le \phi 1.5^k|V_k| = \phi\left(\frac{1.5}{2}\right)^k|V_0| = \phi 0.75^k|V_0|.$$

■

The simple separator lemma is interesting when we consider the case in which $k = O(\log n)$. In this case, the separator for G_0 is bounded by

$$|S_0| = \phi 0.75^k|V_0| = \phi 0.75^{\log|V_0|}|V_0| = \phi|V_0|^{\log 0.75}|V_0| \le \phi|V_0|^{0.59}.$$

Thus, even though the separator of G_k contained $O(|V_k|)$ vertices, this same separator when it is projected onto the original graph contains only $O(|V_0|^{0.59})$ vertices. Hence, if k is sufficiently large, a suboptimal separator of G_k does not significantly affect the size of the separator of the graph G_0.

3.3.1 The Nature of a Good Separator

The key element in the proof of the path projection lemma is that the edge-weight of the path in question was average. This is certainly true for any simple path but is it true for a separator path as well?

The answer to this question depends on the algorithm used to compute the vertex separator. In the multilevel algorithm, the vertex separator is computed as being the path along the boundary of the bisection. Since, the bisection is computed so that the number of edges crossing the two parts is minimized, it is not unreasonable to assume that an equal or larger amount of edge-weight does not cross the boundary. Because of this, the separator path obtained from the partition boundary should have on the average at least as much weight as any other path.

Our experimental results verify this observation. In fact, for all coarsening schemes, if we look at the number of vertices as being projected from a coarse graph to the next level finer graph, the increase in the separator size is almost always bounded by 1.5. Hence, assuming that the edge-weight of the separator path is no less than that of any other path, the following lemma is true.

Lemma 3 *The separator of G_0 obtained by projecting the separator of G_k is bounded by $\alpha(1.06)^k\sqrt{|V_0|}$.*

Proof. From the proof of Lemma 2, we have that $|S_0| \leq 1.5^k|S_k|$. From Assumption 1 we know that there is a separator of size $|S_k| \leq \alpha\sqrt{|V_k|}$. Thus,

$$|S_0| \leq 1.5^k|S_k| \leq \alpha 1.5^k\sqrt{|V_k|} \leq \alpha\left(\frac{1.5}{\sqrt{2}}\right)^k\sqrt{|V_0|} \leq \alpha(1.06)^k\sqrt{|V_0|}.$$

■

3.3.2 Effect of Refinement

When a partition of G_k is projected to the next level finer graph G_{k-1}, the algorithm described in Section 2, refines the partition using a scheme that tries to decrease the edge-cut, by moving vertices along the boundary of the cut. This refinement algorithms has two effects. First it tends to further decrease the number of boundary vertices, and second because it minimizes the edge-cut, it increases the edge-weight of the path formed by the vertices along the partition boundary. As a result, even when the coarsening is not perfectly random, by increasing the boundary edge-weight, more vertices can be dropped from the boundary of the G_{k-2} graph.

In our experiments we observed that when refinement is performed, the increase in the number of boundary vertices is usually no more than by a factor of 1.5 and in some cases it is smaller than $\sqrt{2}$. This is why, the separator of graph G_0 in all of our experiments was less than $\alpha\sqrt{n}$.

3.3.3 Experimental Results

To verify the correctness of Lemmas 1, and 3 we performed experiments with three different planar graphs that have triangular faces. As we did throughout the analysis in this section, each vertex $v_i \in V_i$ is treated as a single vertex, irrespective of its weight. Furthermore, the separators were computed as the vertices of the first part that lay along the boundary of the bisection. That is, if V_0 is bisected into A and B, then the set $A' \subset A$ of vertices that are connected to some vertices of B is taken as the separator.

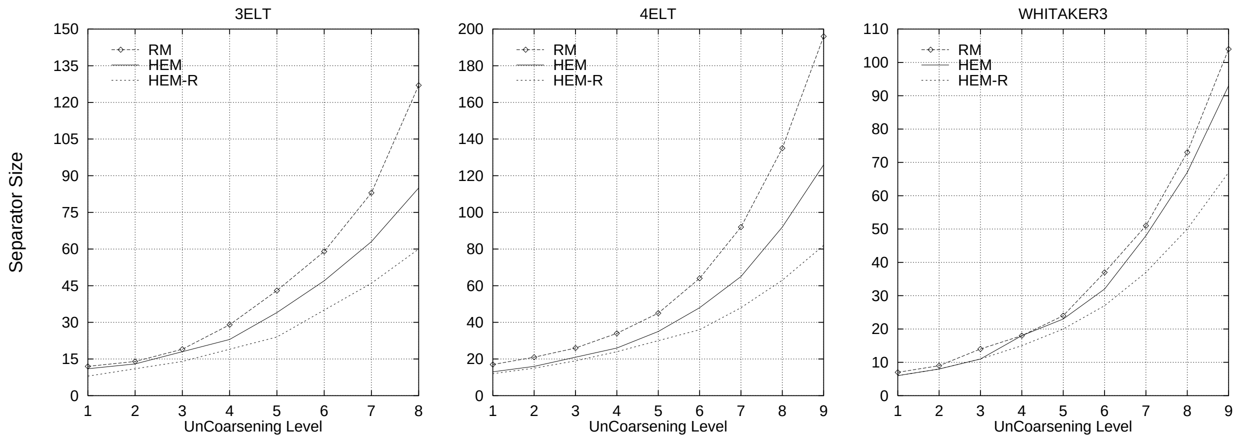

Figure 9: The increase in the number of nodes in the separator at successive uncoarsening levels.

Figure 9 shows how the size of the separator increases at successive coarsening levels. For each matrix, three curves are shown. The two of them, labeled RM and HEM, correspond to random matching and heavy-edge matching with no refinement during the uncoarsening phase, while the one labeled HEM-R, corresponds to heavy-edge with boundary greedy refinement [14]. From this graph, we see that at successive uncoarsening levels, the size of the separator increases by a factor smaller than 2. For example, when RM is used for 4ELT, going from the 7th to the 8th uncoarsening level, the separator increases from 92 to 135 vertices—an increase by a factor of 1.47. Furthermore, comparing RM with HEM, we have that HEM consistently produces smaller separators, which is not surprising, since HEM finds bisections with smaller edge-cuts (Section 3.2). Also, when boundary refinement is used (HEM-R), the size of the final separator is much smaller, and tends to increase at a lower rate as suggested in Section 3.3.2.

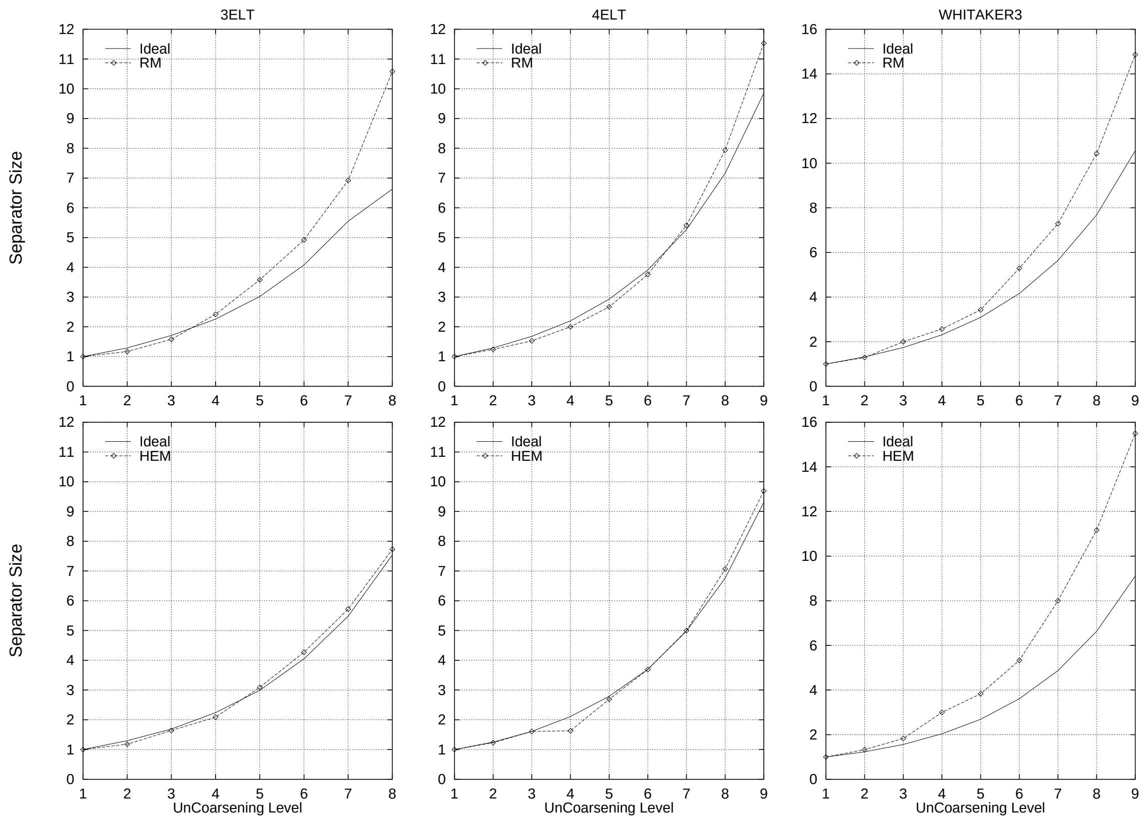

Figure 10: The rate of increase of the separator compared to the ideal increase.

Note that for all the graphs in Figure 9, the size of the separator increases much slower than 1.5 for the first few coarsening levels. This is because the size of the graphs during these last levels does not decrease very fast. For this reason we constructed the graphs shown in Figure 10. In this figure, for each graph and matching scheme we plotted the relative increase of the size of the separator for successive uncoarsening levels, over the size of the initial separator. Also, for each graph and matching scheme we computed the ideal relative increase so that the $\alpha\sqrt{|V_i|}$ bound is maintained. We computed this as follows. If G_k is the coarsest graph then, the ideal separator of G_i is

$$|S_i| \leq \alpha\sqrt{|V_i|} = \alpha\sqrt{|V_k|}\sqrt{\frac{|V_i|}{|V_k|}} = \sqrt{\frac{|V_i|}{|V_k|}}|S_k|.$$

Thus, $|S_i|$ is higher by a factor of $\sqrt{|V_i|/|V_k|}$, and this is the ideal relative increase plotted in Figure 10. Since, RM and HEM lead to coarser graphs that have slightly different number of vertices (*i.e.*, the maximal matching computed by RM and HEM are not of the same size), each matching scheme has a different ideal curve. From these graphs we see that, the overall rate of increase in the size of the separator is worse than the ideal increase. However, the difference is usually very small. The graph for WHITAKER3 is particularly interesting, because both RM and HEM lead to a relatively high increase (a factor of two) in the size of the separator over the ideal increase. The reason for that, is that the initial separator of WHITAKER3 is actually quite small compared to the size of the graph. In particular, the coarsest graph for RM has 88 vertices while the separator has 7, and the coarsest graph for HEM has 118 vertices while the separator has only 6. Consequently, vertices cannot be dropped at the rate dictated by the ideal curve, since it will lead to separators that are contradictory small.

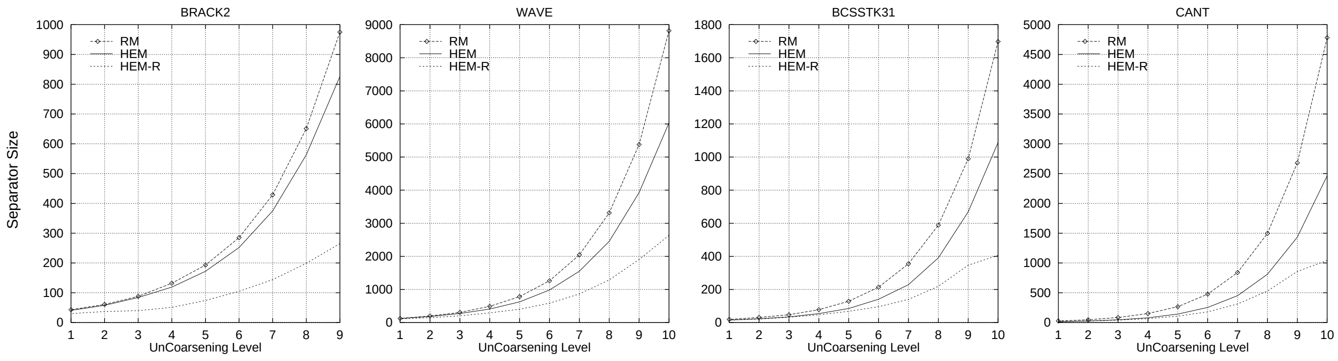

Figure 11: The rate of increase of the separator for 3D finite element meshes.

Finally, Figure 11 shows how the size of the separator increases at successive uncoarsening levels for graphs that correspond to 3D finite element meshes. As it was the case for planar graphs, the size of the separator decreases by a factor smaller than two at each successive uncoarsening level. Also, HEM finds smaller separators at the coarsest graph, and the size of the separator increases at a slower rate than RM. Also, in the case of HEM-R, the size of the separator increases very slowly. For 3D graphs, the ideal increase of the separator size should be $2^{0.75} \approx 1.68$ at each successive uncoarsening level. From these graphs, we that the rate of increase is usually higher than that by a small factor. For instance, in the case of BCSSTK31 and RM, going from the 9th to the 10th uncoarsening level, the separator increased from 989 vertices to 1698 vertices, an increase by a factor of 1.72.

4 Concluding Remarks

The analysis presented in this paper shows that a good partition of the coarsest graph leads to a reasonably good partition of the finer graph, provided the coarsening scheme is a reasonable one. In the multilevel algorithm, this projected partition of the finer graph is actually even better than predicted because refinement during the uncoarsening phase further improves it. Hence the overall partition computed by multilevel scheme is often quite close to the optimal partition.

It may appear that the random matching (RM) scheme does not use the information about the structure of the graph, as its name implies that it is random. But in RM, two vertices are collapsed only if they are connected. Thus, even RM uses information about the structure of the graph during coarsening. HEM does even better, as it tries to keep densely connected set of vertices together.

Note that it is possible to find matching schemes that will lead to arbitrarily poor coarsening method (and poor performance). One such matching scheme is one in which each vertex v is matched with another vertex u (irrespective of whether v and u are connected via an edge). Such a matching scheme will lead to a very poor coarsening scheme, and poor performance overall. Our analysis does not even apply to such a scheme, as this scheme will destroy the structure of the graph (destroy planarity in 2D case).

References

[1] Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings of the sixth SIAM conference on Parallel Processing for Scientific Computing*, pages 711–718, 1993.

[2] Timothy J. Barth. Aspects of unstructured grids and finite-volume solvers for the euler and navier-strokes equations. In *AGARD Report 787 on Unstructured Grid Methods for Advection Dominated Flows*, pages 6.1–6.60, 1992.

[3] T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[4] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *In Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[5] A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[6] A. George and J. W.-H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, March 1989.

[7] T. Goehring and Y. Saad. Heuristic algorithms for automatic graph partitioning. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, 1994.

[8] Anshul Gupta, George Karypis, and Vipin Kumar. Highly scalable parallel algorithms for sparse matrix factorization. Technical Report 94-63, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1994. Submitted for publication in *IEEE Transactions on Parallel and Distributed Computing*. Available on WWW at URL http://www.cs.umn.edu/~karypis/papers/sparse-cholesky.ps.

[9] M. T. Heath, E. G.-Y. Ng, and Barry W. Peyton. Parallel algorithms for sparse linear systems. *SIAM Review*, 33:420–460, 1991. Also appears in K. A. Gallivan et al. *Parallel Algorithms for Matrix Computations*. SIAM, Philadelphia, PA, 1990.

[10] M. T. Heath and P. Raghavan. A Cartesian nested dissection algorithm. Technical Report UIUCDCS-R-92-1772, Department of Computer Science, University of Illinois, Urbana, IL 61801, 1992. To appear in *SIAM Journal on Matrix Analysis and Applications*, 1994.

[11] M. T. Heath and Padma Raghavan. A Cartesian parallel nested dissection algorithm. Technical Report 92-1772, Department of Computer Science, University of Illinois, Urbana, IL, 1992. To appear in *SIAM Journal on Matrix Analysis and Applications*, 1994.

[12] Bruce Hendrickson and Rober Leland. The chaco user's guide, version 1.0. Technical Report SAND93-2339, Sandia National Laboratories, 1993.

[13] Bruce Hendrickson and Rober Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.

[14] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, Department of Computer Science, University of Minnesota, 1995. Also available on WWW at URL http://www.cs.umn.edu/~karypis/papers/mlevel_serial.ps. A short version appears in Intl. Conf. on Parallel Processing 1995.

[15] George Karypis, Anshul Gupta, and Vipin Kumar. A parallel formulation of interior point algorithms. In *Supercomputing 94*, 1994. Available on WWW at URL http://www.cs.umn.edu/~karypis/papers/interior-point.ps.

[16] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.

[17] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

[18] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1979.

[19] J. W.-H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11:141–153, 1985.

[20] G. L. Miller. Finding small simple cycle separators for 2-connected plnar graphs. *Journal of Computer and system Sciences*, 32(3):265–279, June 1986.

[21] Gary L. Miller, Shang-Hua Teng, W. Thurston, and Stephen A. Vavasis. Automatic mesh partitioning. In A. George, John R. Gilbert, and J. W.-H. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithms*. (An IMA Workshop Volume). Springer-Verlag, New York, NY, 1993.

[22] Gary L. Miller, Shang-Hua Teng, and Stephen A. Vavasis. A unified geometric approach to graph separators. In *Proceedings of 31st Annual Symposium on Foundations of Computer Science*, pages 538–547, 1991.

[23] B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. K. Noor, editor, *American Soc. Mech. Eng*, pages 291–307, 1986.

[24] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[25] A. Pothen and C-J. Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software*, 1990.

[26] Alex Pothen, H. D. Simon, and Lie Wang. Spectral nested dissection. Technical Report 92-01, Computer Science Department, Pennsylvania State University, University Park, PA, 1992.

[27] Alex Pothen, H. D. Simon, Lie Wang, and Stephen T. Bernard. Towards a fast implementation of spectral nested dissection. In *Supercomputing '92 Proceedings*, pages 42–51, 1992.

[28] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.

[29] P. Raghavan. Line and plane separators. Technical Report UIUCDCS-R-93-1794, Department of Computer Science, University of Illinois, Urbana, IL 61801, February 1993.