Robust Ordering of Sparse Matrices using Multisection

Cleve Ashcraft[*]   Joseph W.H. Liu[†]

February 9, 1996

Abstract

In this paper we provide a robust reordering scheme for sparse matrices. The scheme relies on the notion of *multisection*, a generalization of bisection. The reordering strategy is demonstrated to have consistently good performance in terms of fill reduction when compared with multiple minimum degree and generalized nested dissection. Experimental results show that by using multisection, we obtain an ordering which is consistently as good as or better than both for a wide spectrum of sparse problems.

1 Introduction

It is well recognized that finding a fill-reducing ordering is crucial in the success of the numerical solution of sparse linear systems. For symmetric positive-definite systems, the minimum degree [38] and the nested dissection [11] orderings are perhaps the most popular ordering schemes. They represent two opposite approaches to the ordering problem. However, they share a common undesirable characteristic. Both schemes produce generally good orderings, but the ordering quality is not uniformly good.

The main contribution of this paper is to introduce a robust ordering scheme that gives good quality orderings consistently, near equal or better than minimum degree and nested dissection. The basic tool is a *multisector*, a generalization of a bisector. A multisector is a subset of vertices whose removal subdivides the graph into two or more components. We call the resulting partition a *domain decomposition*. The whole ordering process has two *independent* phases: the ordering of the vertices in the components (the domains) and the ordering of the multisector (the interface).

An outline of this paper is as follows. In Section 2, we provide evidence on the inconsistent ordering quality of the minimum degree and nested dissection schemes. The multisection ordering scheme is described in Section 3, where the notions of domain decomposition, multisector and multisection ordering are introduced. The quality of the resulting ordering depends on three factors: the domain decomposition, the ordering method for the domains, and the ordering method for the multisector.

In Section 4, we consider numerical experiments of the multisection ordering approach on regular grids. We show that multisection gives an ordering quality close to the optimal nested dissection ordering [11] for square and cubic grids, and local nested dissection ordering [8] for grids of large aspect ratios. Section 5 evaluates multisection on some structural analysis matrices from the Harwell-Boeing collection [10]. We use an

[*]Boeing Information and Support Services, P. O. Box 24346, Mail Stop 7L-22, Seattle, Washington 98124. This research was supported in part by the ARPA Contract DABT63-95-C-0122.

[†]Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant A5509 and in part by the ARPA Contract DABT63-95-C-0122.

incomplete nested dissection scheme to determine a multisector and its associated domain decomposition. Section 6 contains our concluding remarks.

2 Minimum Degree and Nested Dissection Orderings

2.1 General Overview

The *minimum degree ordering* algorithm is a symmetric version of the Markowitz scheme [30]. It was first described and used by Tinney and Walker [38]. The basic minimum degree algorithm can be best described in terms of elimination graphs [35]. Let G be the graph associated with the given sparse matrix. The scheme selects a vertex v of minimum degree in G. This vertex is numbered next in the ordering and is eliminated from the graph G to form its elimination graph G_v. The graph G is then replaced by this elimination graph G_v, where the selection/elimination process is repeated. Many important enhancements have been made to the implementation of the minimum degree ordering; the survey paper [14] contains a comprehensive account of such enhancements.

To choose a vertex to eliminate, the minimum degree algorithm uses the degree of the vertex, which is a local graph property. If we view the ordering as the construction of the elimination tree[1], the tree is formed from bottom-up. This means nodes associated with the bottom part of the tree get their ordering assignments first.

The minimum degree ordering has been generally recommended as a general purpose fill-reducing reordering scheme. Its wide acceptance is largely due to its effectiveness in reducing fill and its efficient implementation. However, since the scheme uses only local information, it produces only adequate orderings for large problems. There is room for improvement.

Another popular fill-reducing ordering is the *nested dissection ordering* [11] and its generalizations. Contrary to the minimum degree algorithm, nested dissection is a top-down scheme. It uses the notion of a *separator*, which is a subset of nodes whose removal renders the remaining graph disconnected. Nested dissection finds a separator and numbers the nodes in the separator last. The process is then repeated recursively on each component. Nested dissection constructs the elimination tree from the root down.

The theoretical foundation of nested dissection was given by George [11], and he showed that nested dissection on grid problems will give optimal orderings with respect to factor nonzeros and factorization operation counts. For general graph problems, the recursive approach of finding and ordering separators is often referred to as *generalized nested dissection*. For the remainder of this paper, we shall simply use nested dissection to refer to generalized nested dissection.

A separator is a global property of the graph. The quality of a nested dissection ordering depends crucially on the quality of its separators. With good separators, nested dissection gives high quality orderings for a large class of graphs. Unfortunately, that is not the case for some type of graphs, for example, those of large aspect ratios. A good separator is not enough; the ordering of the vertices found in the different levels of the separators is important. In many cases, the ordering given by the recursive ordering of the separators in nested dissection can be improved.

[1] The elimination tree is a useful tool in the study of sparse factorization. For more details, see [29].

2.2 Shortcomings of Minimum Degree and Nested Dissection

To illustrate the inconsistent ordering quality from the minimum degree (MMD) and the nested dissection (ND) algorithms, we apply the two orderings on a sequence of rectangular grids of increasing aspect ratios. We run the orderings on grids of the following sizes: 128×128, 256×64, 512×32, 1024×16, 2048×8, and 4096×4 respectively, so that the number of unknowns are the same ($2^{14} = 16384$).

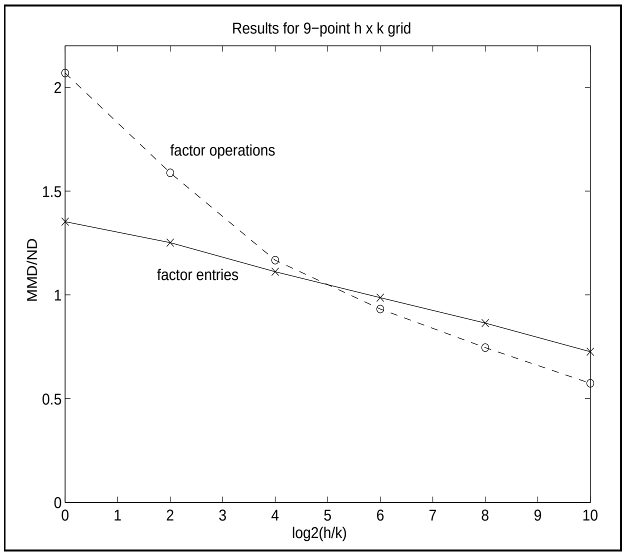


FIG. 1. *Comparison of* MMD *and* ND *for 2-D regular grids of different aspect ratios.*

In Figure 1, we plot the performance ratio of MMD/ND versus the base-2 logarithm of the aspect ratios of the rectangular grids. The variation in performance is rather drastic. For the grid of unit aspect ratio, ND outperforms MMD by a factor of two in factorization operations. On the other end, for grids of large aspect ratios, MMD is better than ND in operations by a factor of close to two.

Such performance variations can also be found in practical sparse matrix problems. In Section 5 we will compare minimum degree, nested dissection and multisection on a set of matrices from the Harwell-Boeing sparse matrix collection [10]. For seven of the sixteen matrices minimum degree generates an ordering with fewer operations than our nested dissection ordering. Part of this can be explained by the aspect ratio of the graphs of the matrices. For example, one of the test matrices, BCSSTK25, is the finite element model of a tall building and has a graph with a large aspect ratio.

The shortcomings of the minimum degree ordering are largely due to the local nature of the algorithm. Making a node selection based on the local degree information often can lead to less-than-desirable choices. Berman and Schnitger [7] have shown that there is a minimum degree sequence for the square grid so that the resulting ordering has factor nonzeros and operation counts an order of magnitude more than the optimal. By construction, their less-than-optimal ordering gives separators with a severe "fractal" nature. This property is found to a lesser degree in virtually any minimum degree ordering.

On the other hand, the shortcoming of the nested dissection can be best explained by its performance on problems of large aspect ratios. The experimental results in Figure 1 on rectangular grids of varying aspect ratios, show that the difference in performance is quite significant. Since we are using the best separator (best in terms of both the separator size and the component balance) on the grid at each step of nested dissection, we cannot attribute the problem to the quality of the separators. The problem is with the way the last few levels of separators are numbered. Indeed, our approach of using multisectors provides a more effective way of numbering the nodes associated with these separators.

3 The Multisection Ordering Algorithm

In nested dissection, a separator in the form of a *bisector* is used that splits the given graph into two subgraphs, where each subgraph is ordered recursively. The nodes associated with the bisector usually form a clique in the filled graph. Our approach uses the notion of a *multisector* separator, which splits the given graph into a number of subgraphs. In general, the multisector nodes induce a sparse submatrix in the filled graph. Within this framework, we can therefore view nested dissection as a *multi-level bisection* scheme. On the other hand, this new approach is a *bi-level multisection* scheme. For simplicity, we shall simply refer to it as *multisection*.

3.1 Domain Decomposition and Multisector

This approach is closely related to the notion of domain decomposition, which we now formally define. Let A be a matrix with symmetric structure and let $G = (V, E)$ be an undirected graph where V are vertices and $E \subseteq V \times V$ are the edges. Edge (i, j) is in E if and only if $a_{i,j} \neq 0$. For a subset $U \subseteq V$, the *boundary* of U, written $\mathit{Adj}(U)$, is the set of all vertices adjacent to those in U but not including any in U, i.e., $\mathit{Adj}(U) = \{v \notin U \mid (u, v) \in E \text{ for some } u \in U\}$.

Consider a block partition of the vertex set V:

$$V = \Phi \cup \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_M,$$

where each Ω_i is a *domain* and Φ is the set of *interface* vertices. Each domain Ω_i is a connected subgraph of G whose boundary $\mathit{Adj}(\Omega_i)$ is contained in Φ. Since the domains are separated from one another by the set Φ, we shall also refer to Φ as a *multisector*, for it generalizes the notion of a bisector. Without loss of generality, we shall assume that the multisector partition is nontrivial; that is, $M > 1$ and Φ is nonempty.

Given this domain decomposition, we impose one condition on the ordering: *All vertices in the domains are numbered before any vertex in the multisector.* Since the domains are isolated from each other, each can be ordered independently. For domain Ω_i, we construct the graph $G_i = (\Omega_i, E_i)$ where $E_i = E \cap (\Omega_i \times (\Omega_i \cup \mathit{Adj}(\Omega_i)))$ contains all edges $(u, v) \in E$ where $u \in \Omega_i$. To order the multisector vertices we define the set

$$\overline{\Phi} = V \setminus \Phi = \bigcup_i \Omega_i,$$

that is, the set of domain vertices. We use the conventional notation $G_{\overline{\Phi}}$ to represent the *elimination graph* of G after all vertices in the domains have been eliminated. Note that $G_{\overline{\Phi}}$ is also the graph of the *Schur complement matrix*

$$A_{\Phi,\Phi} - \sum_{i=1}^{M} A_{\Phi,\Omega_i} A^{-1}_{\Omega_i,\Omega_i} A_{\Omega_i,\Phi}.$$

From this equation it is clear that the elimination graph $G_{\overline{\Phi}}$ (the structure of the Schur complement matrix) does not depend on the ordering of domain vertices. Therefore, the ordering of the multisector vertices in this elimination graph can proceed independent of the ordering of the domain vertices.

Figure 2 is an example of a domain decomposition of a 6×6 grid graph. The vertices are partitioned into six domains and a multisector:

$$\Phi = \{2, 4, 6, 7, 8, 10, 14, 16, 18, 19, 20, 21, 22, 23, 27, 33\}.$$

Multisector vertices are represented by squares and domain vertices by circles.

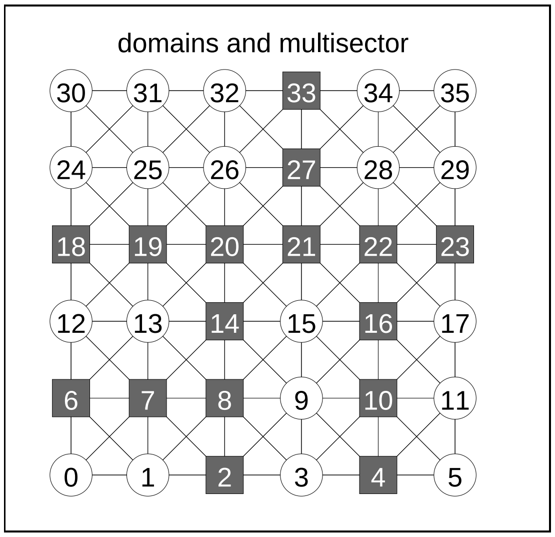


FIG. 2. *A domain decomposition example.*

A useful notion related to domain decomposition is the idea of *indistinguishable vertices.* For a given graph, two vertices are said to be *indistinguishable* if they are adjacent and have exactly the same set of neighbors (except themselves). The elimination graph $G_{\overline{\Phi}}$ of a domain decomposition usually has many fewer indistinguishable vertices than vertices. This is important since the execution time and, to some extent, the quality of the minimum degree ordering depends on the number of indistinguishable vertices instead of the number of original vertices. Figure 3 contains the Schur complement matrix associated with the domain decomposition of Figure 2. There are 16 vertices and 10 indistinguishable ones. For example $\{4, 10, 16\}$ forms an indistinguishable set, since they all have the same adjacent set $\{2, 8, 14, 20, 21, 22, 23\}$ in the elimination graph.

3.2 The Multisection Ordering Algorithm

Immediately below is a skeleton of the multisection ordering scheme based on the notion of a domain decomposition.

```
MS (Ordering method-1, Ordering method-2):
    Given a domain decomposition $(\Phi, \Omega_1, \Omega_2, \ldots \Omega_M)$ of $V$ ;
    **for** each domain $\Omega_i$ **do**
        Order the graph $G_i = (\Omega_i, E_i)$ by ordering method-1 ;
    Form the elimination graph $G_{\overline{\Phi}}$ and order by ordering method-2 ;
```

A multisection ordering is defined by three choices:

1. How to determine the domain decomposition?

FIG. 3. *The Schur complement matrix has 16 vertices, 10 indistinguishable vertices, original entries 'x', fill entries '+'*

$$
\begin{array}{c|c|ccc|cc|cc|cc|c|c|c|c|cc|}
 & 2 & 4 & 10 & 16 & 6 & 7 & 8 & 14 & 18 & 19 & 20 & 21 & 22 & 23 & 27 & 28 \\
\hline
2 & \times & + & + & + & + & \times & \times & + & & & + & + & + & & & \\
\hline
4 & + & \times & \times & + & & & + & + & & & + & + & + & + & & \\
10 & + & \times & \times & \times & & & + & + & & & + & + & + & + & & \\
16 & + & + & \times & \times & & & + & + & & & + & \times & \times & \times & & \\
\hline
6 & + & & & & \times & \times & + & + & + & + & + & & & & & \\
7 & \times & & & & \times & \times & \times & \times & + & + & + & & & & & \\
\hline
8 & \times & + & + & + & + & \times & \times & \times & + & + & + & + & + & & & \\
14 & + & + & + & + & + & \times & \times & \times & + & \times & \times & \times & + & & & \\
\hline
18 & & & & & + & + & + & + & \times & \times & + & + & & & + & + \\
19 & & & & & + & + & + & \times & \times & \times & \times & + & & & + & + \\
\hline
20 & + & + & + & + & + & + & + & \times & + & \times & \times & \times & + & & + & + \\
\hline
21 & + & + & + & \times & & & + & \times & + & + & \times & \times & \times & + & \times & + \\
\hline
22 & + & + & + & \times & & & + & + & & & + & \times & \times & \times & \times & + \\
\hline
23 & & + & + & \times & & & & & & & & + & \times & \times & + & + \\
\hline
27 & & & & & & & & & + & + & + & \times & \times & + & \times & \times \\
28 & & & & & & & & & + & + & + & + & + & + & \times & \times \\
\hline
\end{array}
$$

2. What fill-reducing ordering to use to order the domains Ω_i?

3. What fill-reducing ordering to use to order the multisector?

In the literature, there are a number of existing ordering schemes using this multisection approach in MS.

- One-Way Nested Dissection — MS(PROFILE, PROFILE)

  The *one-way dissection* scheme by George [12] chooses a set of parallel dissectors as its multisector. Each component in the remaining graph is ordered by a profile ordering. The elimination graph associated with the multisector is also numbered by a profile ordering. We can therefore view one-way dissection as a MS(PROFILE, PROFILE) multisection ordering scheme. George provided experimental and theoretical results to show that one-way dissection can be better than nested dissection for grid graphs with large aspect ratios. In his master thesis [31], Ng has considered the recursive use of the one-way dissection approach. This can also be viewed as using some form of multisector.

- Local Nested Dissection — MS(ND, BAND)

  The *local nested dissection* LND scheme in [8] carries the one-way dissection idea further. A rectangular grid of large aspect ratio is subdivided into roughly square domains by a set of parallel horizontal and vertical dissectors. Each square domain is ordered by nested dissection. The multisector defined by the set of parallel dissectors is then numbered by a band ordering. The local nested dissection scheme is therefore a MS(ND,BAND) multisection ordering method.

- Incomplete Nested Dissection — MS(CMD, ND) and MS(MMD, ND)

  There are two generic forms of an *incomplete nested dissection* ordering. In both forms, the multisector is constructed using the recursive bisection process of nested dissection and the ordering of the multisector vertices follows the given nested dissection ordering. The difference lies in how the vertices in the domains are ordered, using either MMD, multiple minimum degree [26] or CMD, constrained minimum degree [28]. The latter algorithm uses the multisector nodes to compute the degrees of nodes in the domains and so usually generates a better ordering than the former on the domain subgraphs. There are many examples of incomplete nested dissection in the literature [3], [5], [6], [9], [16], [17], [20], [22], [25], [27], [33], [34], including two excellent state-of-the-art software packages, CHACO from Sandia National Laboratories [18] and METIS from the University of Minnesota [21].

The above methods are all members of the multisection family of ordering algorithms. In the following sections we will compare incomplete nested dissection algorithms with a new method, MS(CMD, MMD), where vertices in the domains are ordered with constrained minimum degree and the Schur complement graph is ordered with multiple minimum degree. We will refer to the MS(CMD, MMD) algorithm as *multisection*, abbreviated MS, in contrast with an incomplete nested dissection method, abbreviated as ND.

What remains is to specify how the multisector is created. In [5], the authors looked at two possibilities. The first method is to order the graph using MMD and use the elimination tree to extract a multisector. (Each domain is a subtree of the elimination tree; the multisector consists of all remaining vertices.) This multisector is then *smoothed* to remove the fractal nature of the separators that form the multisector. The second method, as described in [3], [5] performs recursive bisection on the graph until the subgraphs are a certain size, then take the multisector to be the union of the separators.

Subsequent to [5], Rothberg independently discovered the multisection method using the second technique [36]. In his work, a multisector was formed of the separators obtained from the CHACO code [18] and automatic nested dissection from SPARSPAK [13]. Multisection consistently performed as well or better than the nested dissection algorithm that generated the multisector, evidence that supports our results in Section 5 where we obtain a multisector from the METIS software package [21] and our own nested dissection software [4].

4 Experimental Results on Regular Grids

In this section we present some experimental results for the MS(CMD,MMD) ordering algorithm on regular grids. For grid problems it is easy to construct an ideal set of multisectors. In this way, we can study the effectiveness of MS(CMD,MMD) ordering when compared with some theoretically-optimal orderings.

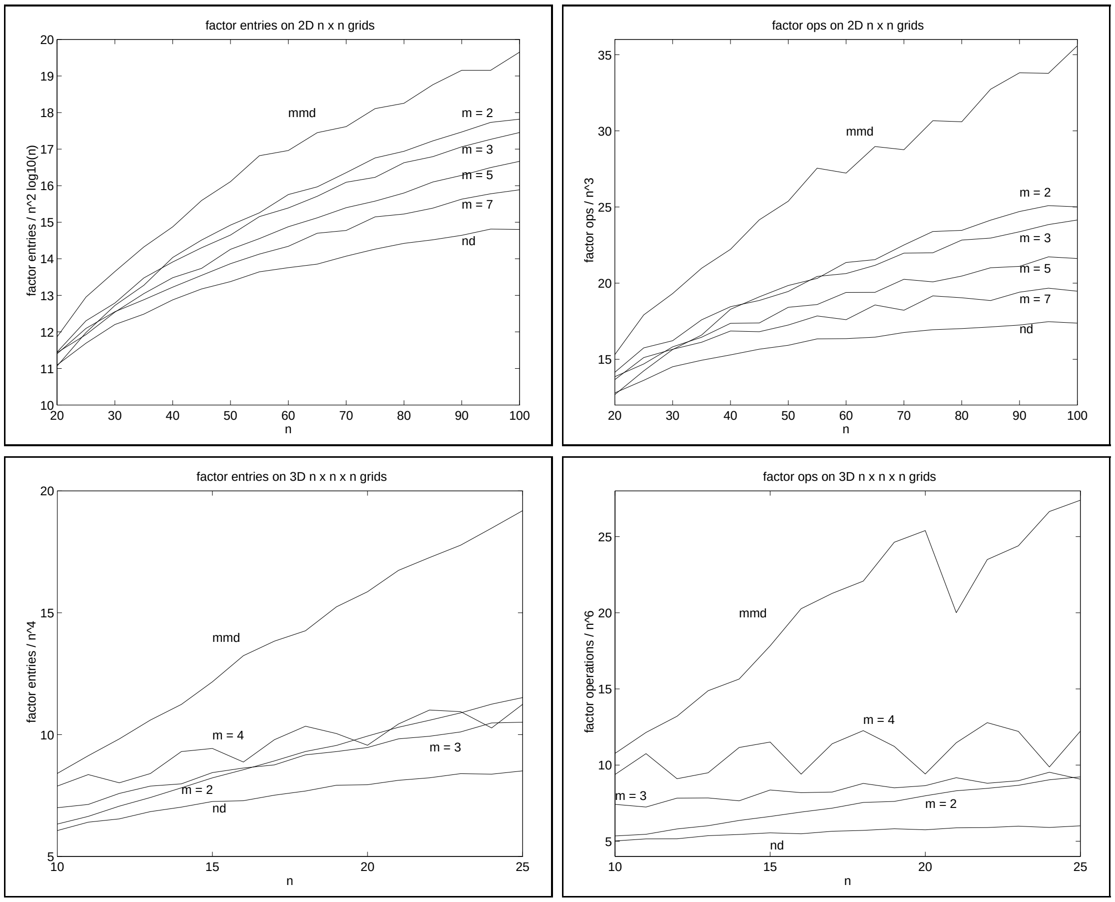
4.1 Square and Cubic Regular Grids

We first consider 9-point operators on 2-D $n \times n$ grids and 27-point operators on 3-D $n \times n \times n$ grids. For these graphs, George's nested dissection ordering [11] gives the best results (aside from some minor variations on very small grids due to edge effects). For a square $n \times n$ grid, nested dissection first bisects the grid with a vertical separator creating two subgrids, each approximately $n/2 \times n$ in size. Each of these subgrids is bisected by a horizontal separator forming a total of four $n/2 \times n/2$ subgrids. The process repeats on each of the subgrids recursively. The separators are vertical or horizontal lines of grid points.

We construct a multisector in a similar way, composed of horizontal and vertical grid lines (planes in 3-D) that span the grid. The simplest multisector we call Φ_2 which splits the grid into four components in 2-D (eight components in 3-D) with a single separator in each grid direction. Φ_2 splits each side of the grid into two, so there are four domains in 2-D, each approximately $n/2 \times n/2$ in size. For a 3-D grid there are eight domains, each approximately $n/2 \times n/2 \times n/2$ in size. The multisector Φ_3 splits the 2-D grid into nine domains, each approximately $n/3 \times n/3$ in size. In general, the Φ_m multisector creates $M = m^2$ domains in 2-D, ($M = m^3$ domains in 3-D), each domain is roughly n/m along each side. Note, the multisector Φ_1 is empty, the entire grid is one domain. We can parameterize the MS(CMD,MMD) ordering by the Φ_m multisector, and Φ_1 corresponds to MMD on the original grid graph.

Our first experiment is to fix the number of domains and let the grid sizes grow. For 2-D $n \times n$ grids we looked at $1 \leq m \leq 7$. For 3-D $n \times n \times n$ grids we looked at $1 \leq m \leq 4$. Figure 4 contains four plots, results for 2-D grids at the top, 3-D grids on the bottom. The ratio of MS(CMD,MMD) factor entries to those of nested dissection are found on the left, factorization operations on the right.

FIG. 4. *Multisection vs nested dissection on square and cubic regular grids*



For ND, the number of factor entries is $O(n^2 \log n)$ in 2-D and $O(n^4)$ in 3-D; the number of factorization operations is $O(n^3)$ in 2-D and $O(n^6)$ in 3-D. We have scaled the number of factor entries and operations appropriately. In each plot the bottom curve is ND while the top curve is MMD. The multisection curves are found between ND and MMD. As n grows, the relative performance of MMD versus ND becomes appreciably worse, more for factorization operations than factor entries and more for 3-D than 2-D.

The difference between MS and ND grows at a much smaller rate. For 2-D grids there is a steady improvement as m increases. For 3-D grids the smaller values of m are better; no doubt this is due to the relatively larger portion of factor entries and operations bound to the top level separators. The difference is mainly a function of n, the *diameter* of the graph, not the number of vertices.

4.2 Rectangular Quadrilaterals and Hexahedra

Multisection performs fairly well when compared to nested dissection on square and cubic regular grids. We now turn to grids with large aspect ratios and compare multisection to LND, local nested dissection [8], the best ordering for rectangular grids.

TABLE 1
Multisection for Rectangular Grids

255×31 grid				
METHOD			NZF	OPS
MMD/LND			1.24	1.73
ND/LND			1.11	1.41
MS/LND				
m_1	m_2	M		
16	2	32	1.14	1.35
24	3	72	1.08	1.22
32	4	128	1.08	1.20
40	5	200	1.08	1.23
48	6	288	1.08	1.25

$127 \times 15 \times 15$ grid					
METHOD				NZF	OPS
MMD/LND				1.92	3.86
ND/LND				1.12	1.42
MS/LND					
m_1	m_2	m_3	M		
8	1	1	8	1.78	3.44
16	2	2	64	1.12	1.15
24	3	3	144	1.23	1.67

Table 1 presents some results for a 255×31 2-D grid and a $127 \times 15 \times 15$ 3-D grid. Both grids have an aspect ratio of 8, large enough to make local nested dissection clearly better than nested dissection.

The results of MMD, ND and MS are given relative to LND. The performance of ND relative to LND is virtually the same in two and three dimensions; ND requires around 10% more factor entries and 40% more factorization operations. However, MMD shows its strong dependence on the dimensionality of the graph for its 3-D performance is much worse than that for 2-D.

What is important to note is that multisection will generate good orderings with many different multisectors. We have observed this behavior across a wide range of matrices; the quality of the multisection ordering is not strongly dependent on the number of domains induced by the multisector. Furthermore, additional experiments have also shown that the ordering quality is also relatively insensitive to the shape of domains.

5 Experimental Results on Sparse Matrices from Structural Analysis

The experimental results in Section 4 on the 2D and 3D-grid problems suggest that the multisection ordering MS(CMD,MMD) can lead to very competitive orderings. The multisectors used are based on the geometry of the grids, and so can be regarded as the best we can get for a specified number of domains. For general sparse matrix problems, the multisection ordering algorithm depends on the use of an appropriate domain decomposition. To avoid the "fractal" nature of the separators from MMD, each domain should have a "smooth" boundary.

5.1 Finding Domain Decomposition via Recursive Bisection

A simple domain decomposition method can be formulated based on incomplete nested dissection. The vertex set V of the initial graph is decomposed into two or more connected subgraphs by removing a bisector S. The connected components of $V \setminus S$ are recursively bisected until each remaining subgraph is smaller than some prescribed size. The quality of the resulting domain decomposition depends on the method to find bisectors in the recursive steps.

Recently, there have been a number of published papers and software codes that find a partition of a given graph. Notable examples include the CHACO [18] and METIS [21] software packages. We have developed a software code called DDSEP [4], that partitions a graph in three steps.

- (**Initial Multisector**) Construct a multisector that gives a domain decomposition of the graph.
- (**Bisector Constructor**) Apply a block version of the Kernighan-Lin scheme [24] to find a bisector, which is a subset of the multisector.
- (**Bisector Smoother**) Apply a graph matching scheme [27] to improve the bisector.

The DDSEP software has been demonstrated to be quite effective in finding good partitions of general connected graphs and also efficient in terms of execution time [4]. For our purpose of finding domain decompositions for the multisection ordering, DDSEP is appropriate because of its bisector smoothing step in the algorithm. For our experiments, we recursively extract the subgraph under consideration and apply DDSEP to the subgraph. The set of bisectors from the recursive steps collectively forms the multisector. Each un-dissected subgraph is a domain in the domain decomposition.

5.2 Results on Practical Structural Problems

We have selected a set of practical sparse matrices arising from structural analysis problems from the Harwell-Boeing collection [10]. Table 2 provides a list of the problems and their characteristics. The column labeled ORIGINAL contains the number of vertices $|V|$ and number of edges $|E|$ of the original given graph. We have also applied the graph compression technique in [1] to identify the indistinguishable vertices in the original graph. The column labeled COMPRESSED gives the number of vertices $|\mathbf{V}|$ and the number of edges $|\mathbf{E}|$ of the compressed graph. Each compressed vertex $\mathbf{u}$ has a weight $w(\mathbf{u})$, namely the number of vertices in the original graph that $\mathbf{u}$ contains. An edge $(\mathbf{u}, \mathbf{v})$ in the compressed graph has a weight $w(\mathbf{u}, \mathbf{v}) = w(\mathbf{u})w(\mathbf{v})$, and the compressed graph is the smallest graph such that this property holds. All of our ordering software works with the compressed graph, and this oftens results in significantly decreased ordering times when compared with using the original graph.

The last two columns in Table 2 presents the number of factor entries and operations (both additions and multiplications) when the matrices are ordered using our multiple minimum degree software, scaled by 10^3 and 10^6 respectively. For each matrix we made twenty-one runs of MMD where each run randomly permuted the adjacency structure of the graph. Table 2 contains the median values of these runs. When we compare the ordering

TABLE 2
Statistics for Harwell-Boeing Matrices

MATRIX	ORIGINAL $\|V\|$	ORIGINAL $\|E\|$	COMPRESSED $\|\mathbf{V}\|$	COMPRESSED $\|\mathbf{E}\|$	MMD NZF/10^3	MMD OPS/10^6
BCSSTK15	3948	113868	3948	113868	663	172
BCSSTK16	4884	285494	1778	36502	742	146
BCSSTK17	10974	417676	5219	81062	1141	201
BCSSTK18	11948	137142	10926	122177	657	138
BCSSTK23	3134	42044	2930	35256	461	142
BCSSTK24	3562	156348	892	12756	296	38
BCSSTK25	15439	236802	13183	161964	1544	339
BCSSTK29	13992	605496	10202	313846	1721	424
BCSSTK30	28924	2014568	9289	222884	3731	869
BCSSTK31	35588	1145828	17403	288806	5160	2411
BCSSTK32	44609	1970092	14821	226974	5175	1048
BCSSTK33	8738	583166	4344	164284	2656	1300
BCSSTK35	30237	1419926	6611	65934	2782	406
BCSSTK36	23052	1120088	4351	37166	2766	618
BCSSTK37	25503	1115474	7093	88924	2831	558
BCSSTK39	46772	2042522	10140	81762	7671	2194

quality of the nested dissection and multisection methods, we scale their statistics by the corresponding MMD values.

Table 3 contains statistics for two nested dissection algorithms — one from METIS and one using our DDSEP software — and multisection where the multisector has been obtained from either METIS or DDSEP. Again, we made twenty-one runs for each algorithm and matrix and present the median value, scaled by the MMD factor entries (NZF) and factorization operations (OPS). An entry in the table that is greater (less) than one means that the ordering algorithm performed worse (better) than MMD. Both METIS[2] and DDSEP recursively split a subgraph until it has 100 or fewer vertices, (for a weighted graph, until the vertices in the subgraph have total weight less than 100).

Of the two nested dissection algorithms, DDSEP consistently outperforms METIS. We believe that DDSEP produces better separators; see [4] for a comparison. There are three possible reasons.

- DDSEP is more flexible in enforcing a balance constraint than METIS; the latter tries to balance the size of the two subgraphs at the expense of a possible larger bisector. See [37] for a more complete discussion.

- DDSEP works exclusively with vertex bisectors while METIS finds an edge bisector and then extracts a vertex bisector; see [19] for a further explanation.

- DDSEP uses a powerful graph matching algorithm to smooth a bisector.

[2]The options we used for METIS were recommended to us by the author, George Karypis, namely SHEM (sorted heavy edge), BGKLR (combination of boundary greedy and boundary Kernighan-Lin) and GGPKL (graph growing followed by boundary Kernighan-Lin).

Table 3
A comparison of ND *(nested dissection) and* MS *(multisection) relative to* MMD *(multiple minimum degree)*

	NZF				OPS			
	ND		MS		ND		MS	
MATRIX	METIS	DDSEP	METIS	DDSEP	METIS	DDSEP	METIS	DDSEP
BCSSTK15	0.80	0.75	0.83	0.76	0.60	0.53	0.64	0.56
BCSSTK16	1.01	0.97	0.89	0.89	1.01	0.96	0.77	0.77
BCSSTK17	1.07	0.95	0.93	0.86	1.12	0.90	0.80	0.67
BCSSTK18	1.04	0.93	0.91	0.89	0.84	0.77	0.67	0.70
BCSSTK23	1.01	0.84	0.95	0.83	0.88	0.67	0.81	0.66
BCSSTK24	1.18	1.04	1.08	1.00	1.32	1.05	1.13	0.97
BCSSTK25	1.16	1.02	0.96	0.90	1.34	1.14	0.86	0.80
BCSSTK29	1.15	0.96	0.98	0.97	1.15	0.85	0.86	0.89
BCSSTK30	1.29	1.19	1.08	1.05	1.62	1.51	1.16	1.07
BCSSTK31	1.02	0.88	0.90	0.94	0.72	0.55	0.64	0.72
BCSSTK32	1.32	1.08	1.11	0.95	2.00	1.37	1.38	0.87
BCSSTK33	0.93	0.80	0.84	0.78	0.82	0.59	0.68	0.57
BCSSTK35	1.38	1.10	1.14	1.00	2.16	1.33	1.48	0.98
BCSSTK36	1.27	1.07	1.11	0.93	1.68	1.25	1.29	0.82
BCSSTK37	1.35	1.05	1.14	0.92	2.04	1.25	1.45	0.78
BCSSTK39	1.11	0.93	1.02	0.90	1.38	0.94	1.06	0.78
MEAN	1.13	0.97	0.99	0.92	1.29	0.98	0.98	0.79

Both multisection algorithms, the first using the multisector from METIS, the second using the multisector from DDSEP, consistently outperform their corresponding nested dissection algorithms. Where ND is better than MS, the difference is not large. While there are seven matrices where ND using DDSEP does not produce as good an ordering as MMD, there is only one such case for MS using DDSEP.

TABLE 4
Execution time for the ordering algorithms

	ordering CPU time				portion of factorization time			
		ND		MS		ND		MS
MATRIX	MMD	METIS	DDSEP	DDSEP	MMD	METIS	DDSEP	DDSEP
BCSSTK15	1.18	1.33	3.39	3.45	11.7%	13.2%	33.6%	34.2%
BCSSTK16	0.27	2.76	1.40	1.39	3.1%	32.1%	16.3%	16.2%
BCSSTK17	0.82	5.19	4.43	4.45	7.6%	44.0%	37.5%	37.7%
BCSSTK18	2.50	1.81	10.38	10.47	30.9%	22.4%	128.2%	129.3%
BCSSTK23	0.90	0.77	1.95	1.95	10.7%	9.2%	23.2%	23.2%
BCSSTK24	0.07	1.32	0.58	0.57	3.2%	60.0%	26.4%	25.9%
BCSSTK25	3.34	4.79	11.62	11.42	7.1%	10.2%	24.6%	24.2%
BCSSTK29	1.77	6.89	11.22	11.24	7.1%	27.7%	45.1%	45.1%
BCSSTK30	1.66	23.81	11.08	11.15	3.3%	46.6%	21.7%	21.8%
BCSSTK31	4.74	18.51	20.04	20.27	3.3%	13.1%	14.1%	14.3%
BCSSTK32	2.75	28.63	16.89	17.08	4.5%	46.5%	27.4%	27.7%
BCSSTK33	1.04	18.51	5.75	5.78	1.4%	24.2%	7.5%	7.6%
BCSSTK35	0.85	17.69	6.50	6.57	3.6%	74.0%	27.2%	27.5%
BCSSTK36	0.74	13.72	4.20	4.23	2.0%	37.7%	11.5%	11.6%
BCSSTK37	0.93	14.22	6.81	6.82	2.8%	43.4%	20.8%	20.8%
BCSSTK39	1.28	29.99	10.09	10.18	1.0%	23.2%	7.8%	7.8%

Table 4 contains the ordering times for four out of the five methods. All ordering codes are written in C and were run on a Sparc20 using the `gcc` compiler with the `-O4` option. In general, METIS takes modest amounts of CPU times, but it is penalized because it cannot work with the compressed graph.[3] Compare the times for BCSSTK15 where the compressed graph is identical to the original graph. For this matrix, METIS is almost three times as fast as DDSEP. We have observed this general tendency across the entire set of test matrices; DDSEP is usually a factor of two or more slower than METIS on the original graph. Part of this difference is due to the more powerful graph matching smoother, but part is because DDSEP trades more computation for reduced working storage. DDSEP uses only $O(|\mathbf{V}|\log(|\mathbf{V}|))$ working storage and does not replicate or destroy the input structure of the graph.

The ordering times for ND and MS with DDSEP include the time to order the vertices in the domains using our MMD software. The time in MS to order the vertices in the multisector is relatively small, and so we see the ND and MS ordering times are almost identical (again the median of twenty-one runs). It is clear that the bulk of the ordering time is spent finding the multisector via nested dissection.

[3] METIS first finds an edge separator and then extracts the vertex separator using the graph matching algorithm from [32] that does not take into account any vertex weights.

Table 4 also provides the ordering times as the percentage of time needed for the numerical factorization of the MMD ordering. Our multifrontal factorization code from [2] consistently achieves 15-20 mflops for this collection of matrices. The MMD ordering time is a small per cent of the factorization time for the larger problems, while the ND and MS times can be up to five times greater. For all but four matrices, the MS ordering time takes less than one third of the factorization time. All the matrices in Table 4 are small to moderate in size. For larger matrices that we see in practice, up to two million degrees of freedom, the MMD ordering time is a very small fraction of the factorization time, much less than one per cent. The cost of the ND and MS orderings is also negligible.

TABLE 5
The Influence of Maximum Domain Size

		NZF		OPS		CPU	
MATRIX	$\lvert\Omega_{\max}\rvert$	ND	MS	ND	MS	ND	MS
BCSSTK31	100	0.88	0.94	0.56	0.73	20.04	20.27
	200	0.88	0.90	0.57	0.62	17.38	17.49
	400	0.90	0.88	0.59	0.59	15.27	15.29
	800	0.90	0.88	0.61	0.60	13.48	13.62
	1600	0.91	0.88	0.65	0.61	12.08	12.16
BCSSTK37	100	1.05	0.92	1.25	0.78	6.81	6.82
	200	1.07	0.95	1.30	0.85	5.61	5.59
	400	1.08	0.98	1.32	0.94	4.82	4.82
	800	1.12	1.05	1.44	1.12	4.05	4.04
	1600	1.14	1.09	1.54	1.32	3.39	3.39
BCSSTK39	100	0.93	0.90	0.94	0.78	10.09	10.18
	200	0.93	0.89	0.93	0.77	7.88	7.92
	400	0.96	0.91	0.98	0.79	6.66	6.72
	800	0.99	0.92	1.02	0.81	5.56	5.57
	1600	1.01	0.94	1.09	0.85	4.66	4.66

The quality of the ND and MS orderings is somewhat dependent on the depth to which the nested dissection is taken. Table 5 presents some statistics for three of the matrices. We have varied the maximum domain size (domain weight for a compressed graph) that defines when a subgraph will be split. Doubling the maximum domain size roughly means reducing the number of levels in the separator tree by one. The results for BCSSTK31 show that ND improves as one reduces the maximum domain size but the MS ordering becomes worse. For BCSSTK37 both orderings improve as the maximum domain size decreases, while for BCSSTK39 the ordering quality is more flat throughout the parameter range. Note that the bulk of the ordering times are spent finding the separators at the highest levels. In general, the ordering quality of MS tends to be less sensitive to the number of levels of separators that are used to construct the multisector, although this is problem dependent.

6 Concluding Remarks

In this paper, we have introduced *multisection*, a robust ordering method using the notion of multisectors. We have demonstrated that it produces consistently high quality orderings for graphs of different characteristics. Its performance compares favorably with the popular minimum degree ordering MMD, and a state-of-the-art generalized nested dissection software

METIS.

There are several directions for future work. Foremost is to find high quality multisectors at less cost than performing nested dissection. In the past [5] we have found a multisector by first ordering the graph using MMD, extracting a multisector from the elimination tree, smoothing this multisector, and then ordering using multisection, for a total cost of between two and three times a single MMD ordering. In general, the quality of these orderings lie somewhere between those of MMD and MS using nested dissection to find the multisector. We have constructed decent multisectors using the *generalized pseudo-extents* algorithm from [15]. Again, the ordering quality lies between that of MMD and MS using nested dissection to find the multisector, but finding the multisector can take considerable time, particularly when the number of domains is large. Each of these two methods produce multisectors that are locally smooth, i.e., the boundary of each domain is smooth, but there is little or no *global* smoothness as is found in a multisector from nested dissection. In other words, the multisector may not contain a good *global* bisector that can be easily found by the MMD ordering.

It appears that some type of global smoothness should be present, i.e., the multisector must contain smooth portions at a higher level than the boundary of a single domain. We feel that a viable approach is to form the multisector using recursive *multisection* of the graph, in the same spirit as the quadrisection and octasection from [17] and the k-way partitioning from [23]. This has the potential to reduce the ordering time because fewer levels of recursion are required, and possibly improve the resulting ordering, for often a multisector with a small number of subgraphs has better properties than the equivalent multisector found by repeated application of bisection.

Acknowledgements.

We would like to thank Ed Rothberg and Bruce Hendrickson for much helpful correspondence during the course of this research.

References

[1] C. Ashcraft. Compressed graphs and the minimum degree algorithm. *SIAM J. Sci. Comput.*, 16:1404–1411, 1995.

[2] C. Ashcraft, R. G. Grimes, and J. G. Lewis. Accurate symmetric indefinite linear equation solvers. Technical Report ISSTECH-95-029, Boeing Computer Services, 1995.

[3] C. Ashcraft and J. W. H. Liu. A partition improvement algorithm for generalized nested dissection. Technical Report BCSTECH-94-020, Boeing Computer Services, 1994.

[4] C. Ashcraft and J. W. H. Liu. Using domain decompositions to find graph bisectors. Technical Report ISSTECH-95-024, Boeing Computer Services, 1995.

[5] C. Ashcraft and J. W. H. Liu. Generalized nested dissection: some recent progress. Minisymposium presentation at the Fifth SIAM Conference on Applied Linear Algebra, Snowbird, Utah, June 18, 1994.

[6] S. T. Barnard and H. 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.

[7] P. Berman and G. Schnitger. On the performance of the minimum degree ordering for Gaussian elimination. *SIAM J. Matrix Analysis and Applic.*, 11:83–88, 1990.

[8] M. V. Bhat, W. G. Habashi, J. W. H. Liu, V. N. Nguyen, and M. F. Peeters. A note on nested dissection for rectangular grids. *SIAM J. Matrix Analysis and Applic.*, 14:253–258, 1993.

[9] T. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *Proceedings of Sixth SIAM Conference on Parallel Processing*, pages 445–452, 1993.

[10] I.S. Duff, R.G. Grimes, and J.G. Lewis. Sparse matrix test problems. *ACM Trans. on Math. Software*, 15:1–14, 1989.

[11] J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[12] J. A. George. An automatic one-way dissection algorithm for irregular finite element problems. *SIAM J. Numer. Anal.*, 17:740–751, 1980.

[13] J. A. George, J. W. H. Liu, and E. G. Ng. User's guide for SPARSPAK: Waterloo sparse linear equations package. Technical Report Tech. Rep. CS78-30(revised), Dept. of Computer Sciences, Univ. of Waterloo, Waterloo, Ontario, Canada, 1980.

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

[15] T. Goehring and Y. Saad. Heuristic algorithms for automatic graph partitioning. Technical report, Computer Science Department, University of Minnesota, Minnesota, 1995.

[16] M.T. Heath and P. Raghavan. A cartesian nested dissection algorithm. Technical Report UIUCDCS-R-92-1772, Dept of Computer Science, University of Illinois, Urbana, IL, 1992.

[17] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, Sandia National Laboratories, Albuquerque, NM, 1992.

[18] B. Hendrickson and R. Leland. The Chaco user's guide. Technical Report SAND93-2339, Sandia National Laboratories, Albuquerque, NM, 1993.

[19] B. Hendrickson and E. Rothberg. A multi-level approach to computing node separators for nested dissection ordering. Technical report, Sandia National Laboratories, 1996. in progress.

[20] 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, Minnesota, 1995.

[21] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, Minnesota, 1995.

[22] G. Karypis and V. Kumar. Multilevel graph partition and sparse matrix ordering. In *Intl. Conf. on Parallel Processing*, 1995.

[23] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical report, Department of Computer Science, University of Minnesota, Minnesota, 1995.

[24] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[25] C. E. Leiserson and J. G. Lewis. Orderings for parallel sparse symmetric factorization. In *Parallel Processing for Scientific Computing*, pages 27–31, 1989.

[26] J. W. H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Trans. on Math. Software*, 11:141–153, 1985.

[27] J. W. H. Liu. A graph partitioning algorithm by node separators. *ACM Trans. on Math. Software*, 15:198–219, 1989.

[28] J. W. H. Liu. On the minimum degree ordering with constraints. *SIAM J. Sci. Stat. Comput.*, 10:1136–1145, 1989.

[29] J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM J. Matrix Analysis and Applic.*, 11:134–172, 1990.

[30] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Sci.*, 3:255–269, 1957.

[31] E. Ng. On one-way dissection schemes. Master's thesis, Dept of Computer Science, University of Waterloo, Waterloo, Ontario, 1979.

[32] A. Pothen and C. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. on Math. Software*, 16:303–324, 1990.

[33] A. Pothen, H. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Analysis and Applic.*, 11:430–452, 1990.

[34] P. Raghavan. Parallel ordering using edge contraction. Technical Report CS-95-293, Dept. of Computer Science, The University of Tennessee, Knoxville, Tennessee, 1995.

[35] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. Read, editor, *Graph Theory and Computing*, pages 183–

217. Academic Press, 1972.

[36] E. Rothberg. private communication, 1995.

[37] E. Rothberg. Exploring the tradeoff between imbalance and separator size in nested dissection ordering. unpublished, 1996.

[38] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *J Proc. IEEE*, 55:1801–1809, 1967.