Nested Dissection: A survey and comparison of various nested dissection algorithms

Manpreet S. Khaira   Gary L. Miller   Thomas J. Sheffler

January 1992
CMU-CS-92-106R

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Methods for solving sparse linear systems of equations can be categorized under two broad classes - direct and iterative. Direct methods are methods based on gaussian elimination. This report discusses one such direct method namely Nested dissection. Nested Dissection, originally proposed by Alan George, is a technique for solving sparse linear systems efficiently. This report is a survey of some of the work in the area of nested dissection and attempts to put it together using a common framework.

This research was sponsored by the National Science Foundation under Contract No. CCR-9016641.

Keywords: gaussian elimination, nested dissection, graph separators, fill-in

Contents

1 Introduction

Methods for solving sparse linear systems of equations can be categorized under two broad classes - direct and iterative. Direct methods are methods based on gaussian elimination. This report discusses one such direct method namely Nested dissection. Nested Dissection, originally proposed by Alan George, is a technique for solving sparse linear systems efficiently. There is a lot of literature on subsequent work in this area. This report is a survey of some of the work in the area of nested dissection and attempts to put it together using a common framework. This report also highlights the fact that all the nested dissection algorithms are variations of a single general algorithm, thereby answering the question that is the main goal of this survey namely - *Are the various nested dissection algorithms completely distinct?* Minimization algorithms for the solution of linear systems, which may be viewed equivalently as iterative methods, are beyond the scope of this report but are discussed in [Joh87].

In section 2 we present the matrix approach to gaussian elimination and then show the equivalent graph theoretic version. Band matrices are used as an example to explain some of the basic ideas involved in gaussian elimination. Nested dissection is introduced in section 3. The various nested dissection methods are also presented. The notion of separators and separator trees for graphs is explained. In section 3.6 the idea of Euclidean norm and its connection to fill-in is described. Finally, the various versions of nested dissection methods are shown to be different forms of tree traversal algorithms of a separator tree in section 3.7. Section 4 presents the parallel nested dissection algorithm and compares it with the sequential one.

2 An Overview of Gaussian Elimination

2.1 Gaussian elimination

We are given a system of equations $Mx = b$, where M is an $n \times n$ matrix, x is a vector of variables of length n, and b is a constant vector of length n. In order to find x by Gaussian Elimination two steps need to be performed

1. Reduce M to upper triangular form. (i.e., Find L such that LM is upper triangular)
2. Solve system $LMx = Lb$.

If M is an $n \times n$ symmetric positive definite matrix, the solution process consists of the following two steps

1. Factor M by means of row operations to

$$M = LDL^T$$

   where L is lower triangular and D is diagonal.

2. Solve the systems

$$Lz = b, Dy = z, L^T x = y$$

The amount of time required to factor M using naive methods is $O(n^3)$ and the time required to solve the systems of equations is $O(n^2)$ if M is *dense*. On the other hand, if M is *sparse* (i.e., M contains mostly zero elements) then by avoiding operating on and storing zeros, we may be able to save time and storage space. However, the factorization of M may create non zero entries in L (and L^T) in positions where M contains zeros. The new nonzeros so created are called *fill-in*.

The factorization of M into LDL^T can de described by the following steps. Setting $A_0 = B_0 = M$ we can write

$$
\begin{aligned}
A_0 &= \begin{pmatrix} d_1 & v_1^T \\ v_1 & B_1' \end{pmatrix} \\
&= \begin{pmatrix} 1 & \mathbf{0} \\ v_1/d_1 & I_{n-1} \end{pmatrix} \begin{pmatrix} d_1 & \mathbf{0} \\ \mathbf{0} & B_1' - v_1 v_1^T/d_1 \end{pmatrix} \begin{pmatrix} 1 & v_1^T/d_1 \\ \mathbf{0} & I_{n-1} \end{pmatrix} \\
&= L_1 \begin{pmatrix} d_1 & \mathbf{0} \\ \mathbf{0} & B_1' - v_1 v_1^T/d_1 \end{pmatrix} L_1^T \\
&= L_1 A_1 L_1^T \\
A_1 &= \begin{pmatrix} d_1 & & \mathbf{0} \\ & d_2 & v_2^T \\ \mathbf{0} & & \\ & v_2 & B_2' \end{pmatrix} \\
&= \begin{pmatrix} 1 & & \mathbf{0} \\ & 1 & \\ \mathbf{0} & v_2/d_2 & I_{n-2} \end{pmatrix} \begin{pmatrix} d_1 & & \mathbf{0} \\ & d_2 & \\ \mathbf{0} & & B_2' - v_2 v_2^T/d_2 \end{pmatrix} \begin{pmatrix} 1 & & \mathbf{0} \\ & 1 & v_2^T/d_2 \\ \mathbf{0} & & I_{n-2} \end{pmatrix} \\
&= L_2 A_2 L_2^T \\
& . \\
& . \\
& . \\
A_{n-1} &= D.
\end{aligned}
$$

Here d_k is a positive scalar, v_k is a vector of length $n-k$, and B_k' is an $(n-k) \times (n-k)$ symmetric positive definite matrix. Also, $B_k = B_k' - v_2 v_2^T/d_2$. Hence, finally

$$M = L_1 L_2 \dots L_{n-1} D L_{n-1}^T L_{n-2}^T \dots L_1^T$$

and

$$L = L_1 L_2 \dots L_{n-1}$$

It can be easily shown that

$$L = \sum_{k=1}^{n-1} L_k - (n-2)I$$

We refer to performing the kth step of factorization as *eliminating variable* x_k. Un-eliminated variables x_j and x_k are referred to as being connected if their corresponding off-diagonal components in B_i $(i < j, k)$ are nonzero. As was explained earlier, as the factorization proceeds unconnected variables can become connected (zero elements becoming nonzero i.e., fill-in).

Lemma 2.1 ([Par61]) *The elimination of variable x_k pairwise connects all variables x_i, $i > k$, to which x_k was connected at the point of its elimination.*

Proof: In the equations describing the factorization of M, note that eliminating x_k modifies $B_k^{'}$ by subtracting the rank-one matrix $v_k v_k^T$ from it, forming B_k. The matrix $v_k v_k^T$ has nonzeros in position (i, j) for all i and j corresponding to nonzero components in v_k. Assuming no cancellation in the subtraction, B_k must have nonzeros in the same positions. The above treatment was taken from [Geo73].

2.2 The graph theoretic interpretation

In this section we will try to develop an understanding of gaussian elimination using graph theory. Let Graph$(M) = (V, E)$ be the graph associated with matrix M, such that each variable in the system of equations is associated with a vertex $v_i, i = 1 \ldots n$, and that for each nonzero entry m_{ij} there is an edge (v_i, v_j) with head v_j and tail v_i. Such a graph represents the nonzero structure of the matrix M [Par61]. If M is symmetric, Graph(M) will be an undirected graph. However, if M is not symmetric, $(i.e., m_{ij} \neq m_{ji})$, Graph$(M)$ will have directed edges. We will ignore self loops created by the non zero elements along the principal diagonal of the matrix.

The following definitions describe operations that will prove useful in later sections.

Definition 2.2 *If $G' = (V', E')$ and $G = (V, E)$ are graphs, then $G' \subseteq G$ if $V' \subseteq V$ and $E' \subseteq E$.*

Lemma 2.3 *Graph$(A \times B) = (V', E')$, where $E' \subseteq \{(v, w) | \exists z (v, z) \in$ Graph$(A) \wedge (z, w) \in$ Graph$(B)\}$*

Lemma 2.4 *Graph$(A^{-1}) \subseteq ($Graph$(A))^*$, G^* is the transitive closure of G.*

The above lemma follows easily by using the series expansion of $(I - A)^{-1}$ and noting that the transitive closure of G is the summation of the integral powers of A.

The next section gives an example that explains the definitions and lemmas described in this section more clearly.

Fill-in manifests itself on the graph G as additional edges during the elimination process. Pivoting along a diagonal element in M is equivalent to removal of a vertex v from the graph.

Definition 2.5 *The* deficiency of v, $Def(v)$, *is the set of edges defined by:*

$$Def(v) = \{(u, w) | (u, v) \in E, (v, w) \in E, (u, w) \notin E\}.$$

This represents the set of fill-in edges due to elimination of vertex v.

Definition 2.6 *The graph:*

$$G_v = (V - \{v\}, E(V - \{v\}) \cup Def(v)).$$

is called the v-elimination *graph of G. The* v-elimination *graph is the graph that results from the gaussian elimination of vertex v from the original graph.*

Definition 2.7 *An elimination ordering is a bijection $\alpha : \{1, 2, \ldots, n\} \rightarrow V$ and $G_\alpha = (V, E, \alpha)$ is an ordered graph. This graph may be used as an aid in selecting an elimination ordering that produces minimal fill-in.*

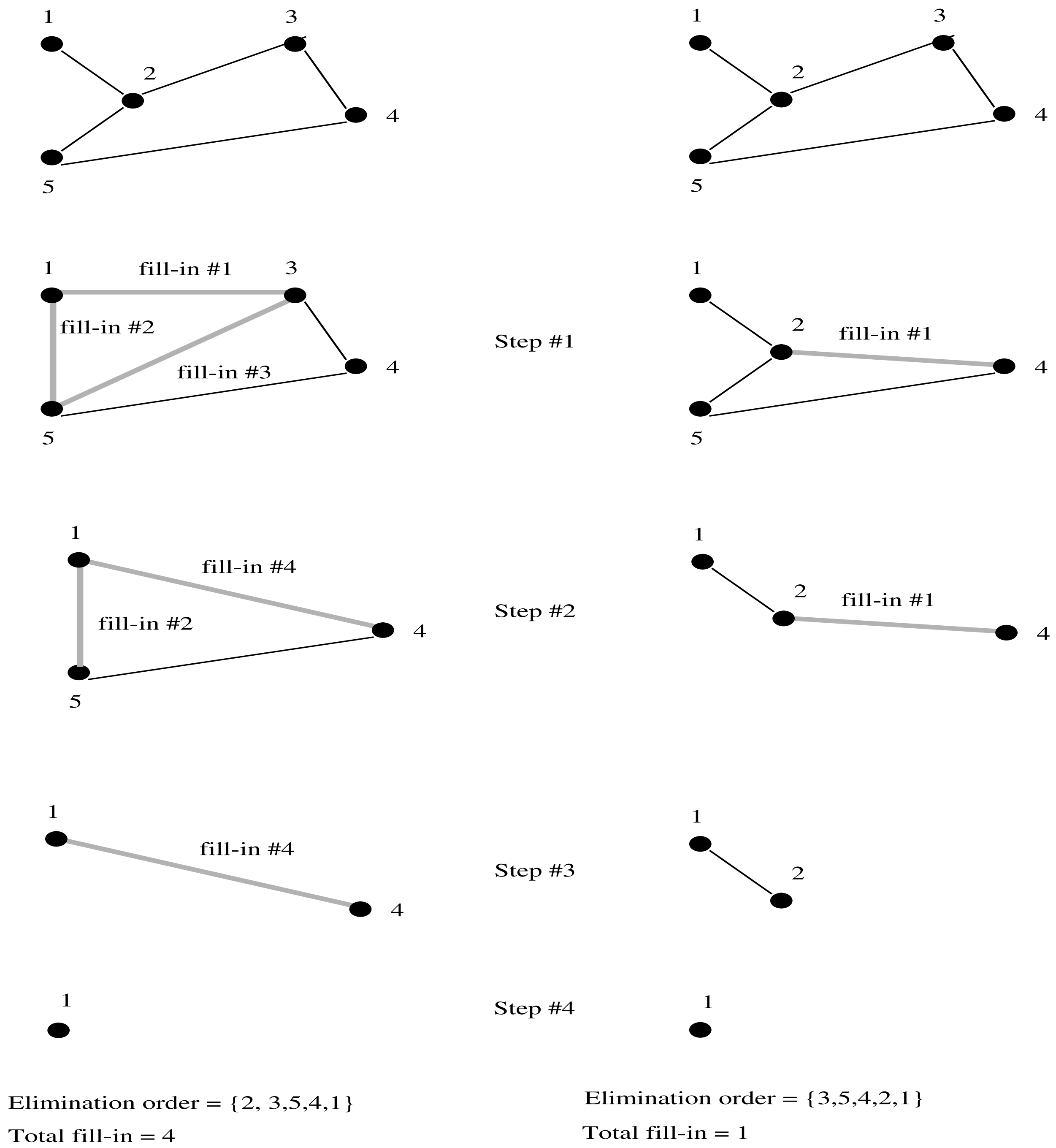


Figure 1: Fill-in with different elimination orders

For an ordered graph, G_α, the elimination process

$$P(G_\alpha) = \{G = G_0, G_1, \ldots, G_{n-1}\}$$

is the sequence of graphs that result from the elimination of the vertices in the order specified by α. The total fill-in is given by

$$F(G_\alpha) = \bigcup_{i=1}^{n-1} Def(v_{\alpha_i})$$

The fill-in that occurs with the elimination of a particular vertex is a function of where that vertex occurs in the elimination ordering α. However, finding an elimination ordering that produces minimum fill-in for a given graph is a problem that has been demonstrated to be $NP-$complete [[GJ79]].

Hence, Reducing a graph $G = G(M)$ to the null graph by successively eliminating vertices $\alpha(1)$, $\alpha(2)$, ..., $\alpha(n)$ is precisely analogous to performing gaussian elimination on matrix M choosing as pivots the diagonal elements that correspond to $\alpha(1)$, $\alpha(2)$, ..., $\alpha(n)$.

Definition 2.8 *A system that may be solved with no fill-in, (i.e., $F(G_\alpha) = \phi$), is called a* monotone transitive graph *or a* perfect elimination graph

It can be observed that the fill-in edges added during gaussian elimination (i.e., $F(G_\alpha)$) on insertion into the original graph G_α will result in a perfect elimination graph $G^\star_\alpha$,

$$G^\star_\alpha = (V, (E \cup F(G_\alpha))).$$

Rose termed this the *monotone transitive extension* of a graph and also characterized these graphs as *triangulated graphs* [Ros72]. A triangulated graph is one in which every cycle of length $n \geq 4$ contains a chord. Figure 1 shows the fill-in resulting from two different elimination orders. Hence, finding a good elimination ordering is essential in reducing the amount of fill-in that occurs during gaussian elimination.

2.3 Band matrices

One application of gaussian elimination that has special properties is that of band matrices. An example where band matrices come up is in the solution of differential equations at discrete points.

Input: $f(x)$.
Goal: $u(x)$ such that $-\frac{d^2u}{dx^2} = f(x), \qquad 0 \leq x \leq 1$.

Because we can add $C + Dx$ (C and D being constants) to the function u and still have the same second derivative, we add two boundary conditions:

$$u(0) = 0, \qquad u(1) = 1.$$

To compute $u(x)$, we break up the interval $[0, 1]$ into equally spaced points $h, 2h, \ldots, nh$ and estimate $u(x)$ at these points using:

$$\frac{d^2u}{dx^2} \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Example: We want to find $u_1, u_2, \ldots, u_6$ where $u_i = u(ih)$ given $f(x)$. Therefore, we need to solve equations of the form:

$$-u_{j+1} + 2u_j - u_{j-1} = h^2 f(jh).$$

Thus, we get the following linear system:

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \\ \vdots \\ u_6 \end{pmatrix} = \begin{pmatrix} f(h) \\ f(2h) \\ f(3h) \\ \\ \vdots \\ f(6h) \end{pmatrix} h^2 \tag{1}$$

The tridiagonal matrix in Equation 1 is very *sparse*, particularly when we take more points. Gaussian elimination could be disastrous if variables are removed in an order that results in a lot of fill-in. Consider the graph of the matrix in Equation 1. For the linear system in Equation 1 the tridiagonal matrix corresponds to the graph in Figure 2.

$$V_1 \rightleftarrows V_2 \rightleftarrows V_3 \rightleftarrows V_4 \rightleftarrows V_5 \rightleftarrows V_6$$

Figure 2: Graph for a band matrix

The problem with applying gaussian elimination on sparse matrices is that, if we are not careful, we can introduce lots of fill (i.e., new nonzero entries). For example, in Figure 3, $*$ represents nonzero entries, $\otimes$ represents zero entries, and $\cdot$ represents zero or non-zero entries in a matrix $M = m_{ij}, i, j = 1 \ldots 5$. If we pivot on m_{11} in order to obtain a zero entry at m_{21} and m_{51}, we may introduce nonzeros at $\otimes$ entries. That is, in Graph(M) we had edges $V_2 \rightarrow V_1$, $V_1 \rightarrow V_3$, and $V_1 \rightarrow V_5$ and, after one row operation to eliminate m_{21}, we may introduce edges $V_2 \rightarrow V_3$ and $V_2 \rightarrow V_5$. Similarly, we may introduced two edges when we eliminate m_{51}.

$$\begin{pmatrix} * & \cdot & * & \cdot & * \\ * & \cdot & \otimes & \cdot & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ * & \cdot & \otimes & \cdot & \otimes \end{pmatrix}$$

Figure 3: Fill introduced by Gaussian Elimination

Let $M = \begin{pmatrix} m_{11} & b \\ c & D \end{pmatrix}$ and let Graph$(M) = (V, E)$. After pivoting on m_{11} we get a new matrix $\begin{pmatrix} m_{11} & 0 \\ 0 & \overline{D} \end{pmatrix}$. Then

$$\text{Graph}(\overline{D}) = (V', E'), \qquad \begin{array}{l} V' = V - \{V_1\} \\ E' \subseteq \{(v, w) | v, w \neq V_1 \wedge ((v, w) \in E \vee (v, V_1), (V_1, w) \in E)\} \end{array}$$

Can we reduce the fill-in by reordering the rows and columns of the band matrix in Equation 1?

Consider:

$$M = \begin{array}{c c} & \begin{array}{cccccc} V_1 & V_3 & V_5 & V_2 & V_4 & V_6 \end{array} \\ \begin{array}{c} V_1 \\ V_3 \\ V_5 \\ V_2 \\ V_4 \\ V_6 \end{array} & \begin{pmatrix} 2 & 0 & 0 & -1 & 0 & 0 \\ 0 & 2 & 0 & -1 & -1 & 0 \\ 0 & 0 & 2 & 0 & -1 & -1 \\ -1 & -1 & 0 & 2 & 0 & 0 \\ 0 & -1 & -1 & 0 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & 2 \end{pmatrix} \end{array}, \qquad A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

In this way we are pivoting on the odd vertices and $S = \{V_1, V_3, V_5\}$. Then, according to the Fill-in Lemma, Graph($\overline{D}$), the fill-in graph, will be a smaller version of the original Graph(M). This can be seen in Figures 5 and 4.

In matrix terms we reduce matrix M to

$$\begin{pmatrix} A & 0 \\ 0 & \overline{D} \end{pmatrix}, \text{ where} \overline{D} = D - CA^{-1}B$$

Recall that the Graph(A^{-1}) is the transitive closure of Graph(A). Then for example, $V_2 \to V_2$ and $V_2 \to V_4$ are in Graph($\overline{D}$) because $V_3 \in S$ and, in the original graph, $V_2 \to (V_3 \to V_3)^* \to V_2$ and $V_2 \to (V_3 \to V_3)^* \to V_4$, respectively, where $(\quad)^*$ represents 1 or more iterations.

$$V_1 \rightleftarrows V_2 \rightleftarrows V_3 \rightleftarrows V_4 \rightleftarrows V_5 \rightleftarrows V_6$$

Figure 4: Original Graph

$$V_2 \rightleftarrows V_4 \rightleftarrows V_6$$

Figure 5: Graph($\overline{D}$)

Generalizing this result for an $n = 2k + 1$ element 3-band matrix, we can reduce the tridiagonal matrix (shown in Figure 6) to the form:

$$\begin{pmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{pmatrix}, \text{ where}$$

$$A = \begin{pmatrix} u_1 & & 0 & \\ & u_3 & & \\ 0 & & \ddots & \\ & & & u_{2k+1} \end{pmatrix}.$$

Therefore, we can pivot on the odd elements and get a fill-in graph half the size of the original graph, as shown in Figure 7.

$$V_1 \rightleftarrows V_2 \rightleftarrows V_3 \rightleftarrows V_4 \rightleftarrows \cdots \rightleftarrows V_{2k}$$

Figure 6: Original tridiagonal matrix graph

$$V_2 \rightleftarrows V_4 \rightleftarrows \cdots \rightleftarrows V_{2k}$$

Figure 7: Graph($\overline{D}$) for a tridiagonal matrix

The dashed lines in Figure 8 shows the edges created by pivoting. These edges are obtained by taking paths starting at a vertex not in A (i.e. an even numbered vertex, V_{even}) to a vertex of A (V_{odd}), then onto vertices of A, and finally to a ending vertex not in A (V_{even}), and replacing the entire path with an edge from the starting vertex to the ending vertex.

$$V_{even} \rightleftarrows V_{odd} \rightleftarrows V_{even}$$

Figure 8: Fill-in detail

Notice that this fill-in corresponds to $D - CA^{-1}B$, where $A = \alpha I$. Thus this reduces to primarily computing CB, and we know computing products of matrices is very expensive. But from a graph theoretic point of view, we are pivoting on a maximum independent set of vertices and, thus, add very little fill-in. That is, when we take paths within vertices of A we are confined to paths using only one vertex. Therefore, finding the elements of CB is equivalent to finding the transitive closure of paths of length 2 and removing the pivot node. This requires a constant amount of work.

3 Nested Dissection

Nested Dissection is a method of finding an elimination ordering. The algorithm uses a *divide and conquer strategy* on the graph. Removal of a set of vertices results in two new graphs on which Gaussian elimination may be performed separately. The results for the two parts may then be combined to find the solution of the entire graph. This method has been shown to result in good elimination orderings for certain classes of graphs.

3.1 Graph separators

A separator of a graph is a relatively small set of vertices whose removal causes the graph to fall apart into a number of smaller pieces. Let S be a class of graphs closed under the subgraph relation (i.e., if $G_2 \in S$ and G_1 is a subgraph of G_2 then $G_1 \in S$). The class S satisfies the *$f(n)$-separator theorem* if there are constants $\alpha < 1, \beta > 0$ such that a separator set with at most $\beta f(n)$ vertices separates the graph into components with at most αn vertices each.

Most algorithms based on separators are recursive, first finding a separator for the whole graph and then finding separators for the components. For these algorithms to work on a graph of class S, all subgraphs of this graph must also be of class S. Hence, the requirement that S be closed under the subgraph relation.

Example: The class of binary trees is closed under the subgraph relation (Why? Separation at any vertex separates the graph into smaller binary trees).

Lemma 3.1 *The class of binary trees satisfies a 1-separator theorem for* $\alpha = \frac{2}{3}$ *and* $\beta = 1$.

A planar graph is one which can be drawn on a plane so that the edges of the graph only intersect at their endpoints. For planar graphs, the following theorem is taken from Lipton and Tarjan [LT80].

Lemma 3.2 ([LT80]) *The class of planar graphs satisfies a* $\sqrt{n}$*-separator theorem for* $\alpha = \frac{2}{3}$ *and* $\beta = 2\sqrt{2}$.

In more recent work, Djidjev proved that the theorem also holds for $\beta = \sqrt{6}$ [Dji81].

These theorems have been presented to provide examples of the types of separators that have been shown to exist, and lead to algorithms for finding separators for limited classes of graphs (i.e., binary trees, planar graphs). Except for the simplest cases, finding separators is a non-trivial problem and no good algorithms exist for finding separators greater than two in size for arbitrary graphs.

3.2 Elimination ordering algorithms

Many variations of elimination ordering algorithms are based on nested dissection. These algorithms have the following basis as a common starting point. The main differences involve the separators found for different classes of graphs and the resulting complexity bounds.

Given a graph G with n vertices, partition G into parts C, A_1, A_2, etc., such that C is a separator of the graph. Number the vertices in C from n down to $(n - |C| + 1)$ so that they are eliminated last from the graph. Recursively number the elements of each of the remaining parts of G $(A_1, A_2, \ldots)$ from 1 to $(n - |C|)$. The procedure continues until all vertices are numbered. Typically, the recursion will cease when the size of a set reaches some small threshold value, n_0, in which case the vertices in the set are arbitrarily assigned numbers in the given range.

3.2.1 Alan George's Nested Dissection Method

The first nested dissection algorithm was proposed by Alan George [Geo73]. His method solves systems whose graphs are $n = k \times k$ square grid graphs in $O(n^{\frac{3}{2}})$ time and $O(n \log n)$ space. George's scheme uses the fact that removal of $O(k)$ ($2k - 1$ precisely) vertices from a $k \times k$ square grid leaves four square grids, each roughly $k/2 \times k/2$.

Example: Figure 9 shows a square grid. Removal of the middle column and middle row (separator set) separates the graph into four subgraphs as explained earlier.

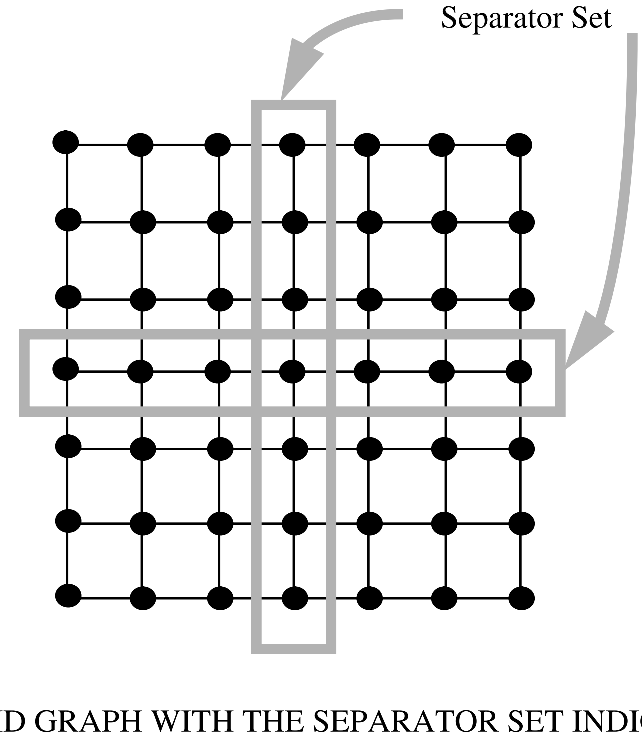


Figure 9: Nested dissection of a grid

The algorithm is as follows. Assume that k is one less than a power of two.

- For $i = 1, \ldots, k$ define
  $\pi(i) = p + 1$ if $i = 2^p(2q + 1)$
  i.e. $\pi(i)$ = number of twos in the prime factorization of i +1
  Also, $\pi(0) = 1$ and $\pi(n) = 1$
- Let $k = 2^l - 1$
  For $m = 1, \ldots, l$ define sets S_m
  $S_m = \{x_{ij} | max(\pi(i), \pi(j)) = m\}$
- Now, number the unknowns (verices) in S_1, followed by those in S_2 and so on, finally numbering the unknowns in S_l (see Figure 10).

Graphs where k is not equal to one less than a power of two may be handled by adding some number of *dummy* vertices. This algorithm results in $O(n \log n)$ fill-in.

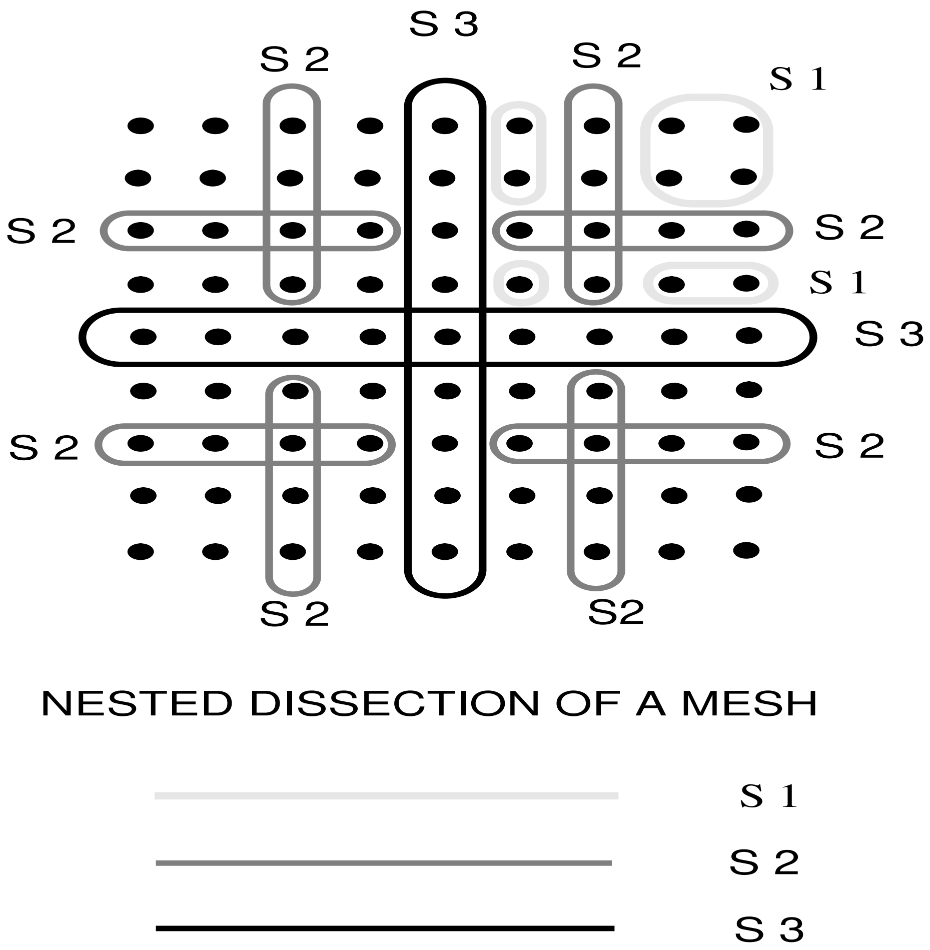


Figure 10: Separator sets in the nested dissection of a grid

Why the method works:
Consider a mesh consisting of k^2 squares called elements, formed by subdividing the unit square $(0,1)\times(0,1)$ into k^2 small squares of side $1/k$, and having a vertex/node at each of the $(k+1)^2$ grid points. With this mesh, we associate the $N \times N$ symmetric positive definite system $Mx = b$, where $N = (k+1)^2$ and each x_i is associated with a node of M. Also,

$M_{ij} \neq 0$ iff x_i and x_j are associated with the nodes of the same element.

In other words, if x_i and x_j are the vertices of the same small square or element then the corresponding matrix component i.e., M_{ij} will be nonzero. However, if there is no element that has both x_i and x_j as vertices then M_{ij} is zero. As an example consider the nested dissection ordering of a 8 $\times$ 8 mesh. Using the algorithm described above the order in which the rows and columns get removed (note removal is not elimination - it is the removal that the ordering algorithm performs) is indicated in the figure. The vertices from sets marked $S3$ subdivide the mesh into 4 subsets which are mutually independent in the sense that if x_i and x_j are in different subsets, then $M_{ij} = 0$ i.e., x_i is not connected to x_j. In the same way vertices in the figure from the $S2$ sets subdivide each of these subsets into 4 subsets which are also mutually independent. As was mentioned in the algorithm, the vertices in the $S3$ sets get the highest elimination ordering numbers. The vertices from $S2$ sets get lower ones and the $S1$ vertices get the lowest ordering numbers. Thus in general the unknowns corresponding to vertices in $S1$ are numbered first (will get eliminated in the gaussian elimination first), followed by those in $S2$ and so on. The way in which the unknowns from a particular set are ordered does not affect the final result. Recall that fill-in will occur i.e., an edge will be inserted between vertices x_i and x_j on removal of a vertex x_k iff both x_i and x_j are connected to x_k. So the elimination of vertices can only cause fill-in within each subset of the set of mutually independent subsets mentioned earlier. This results in a very limited amount of fill-in (for proof refer to [Geo73]).

3.2.2 Generalized Nested Dissection

This algorithm is Lipton, Rose and Tarjan's original version of the generalized nested dissection algorithm [LRT79] . Let S be a class of graphs closed under the subgraph relation on which the $\sqrt{n}$-separator theorem holds. Let α, β be the constants associated with the separator theorem, and let $G = (V,E)$ be an n-vertex graph in S. The recursive algorithm numbers the vertices of G so that sparse Gaussian Elimination is efficient. The algorithm assumes that l of the vertices of G are already assigned numbers, each of which is greater than b (a constant explained later). The goal is to number the remaining vertices of G consecutively from a to b.

- If G contains no more than $n_0 = (\beta/(1-\alpha))^2$ vertices, then number the unnumbered vertices arbitrarily from a to b.
- Otherwise, find sets A, B and C that satisfy the $\sqrt{n}$-separator theorem where C is the separator set. The removal of C divides the rest of G into two components A and B where A and B need not neccessarily be connected components. Let A contain i unnumbered vertices, B contain j and C contain k unnumbered vertices.
- Number the unnumbered vertices in C arbitrarily from b-k+1 to b. In other words, we are assigning the vertices of C the highest numbers.
- Delete all edges whose endpoints are both in C. Apply the algorithm recursively to the subgraph induced by $B \cup C$ to number the unnumbered vertices in B from a=b-k-j+1 to b=b-k. Apply the algorithm recursively to the subgraph induced by $A \cup C$ to number the unnumbered vertices of A from a= b-k-j-i+1 to b=b-k-j.

To begin, call the algorithm with all vertices unnumbered G with a=1, b=n, and l=0. This will number the vertices in G from 1 to n. In this algorithm the vertices in the separator are included in the recursive call but are not renumbered. For any graph all of whose subgraphs satisfy the $\sqrt{n}$-separator theorem, the ordering produced by this algorithm will result in $O(n \log n)$ fill-in and $O(n^{\frac{3}{2}})$ total operation count, although the coefficients of actual fill-in and operation count are very large. However, the authors believe that their worst case bounds are very pessimistic and that the algorithm would be useful for very large graphs.

3.2.3 Gilbert's modification to Generalized Nested Disection

A variation to the generalized nested dissection algorithm described previously has been proposed for separators that divide the graph into more than two pieces [Gil80]. This algorithm assumes that the separator C splits the graph into pieces $A_1, A_2, \ldots, A_r$. A separate recursive call is made for each part, $A_i, 1 \leq i \leq r$.

- If there are no more than n_0 vertices, then simply number the vertices arbitrarily in the range given.
- Find a separator with $k \leq \beta\sqrt{n}$ vertices that divides the graph into connected components $A_1, A_2, \ldots, A_r$, where $|A_i| \leq \alpha n$. Number the vertices of C arbitrarily from $(n - |C| + 1)$ to n.
- Call the algorithm recursively r times for each component A_i, $1 \leq i \leq r$ to number the remaining vertices from 1 to $n - |C|$.

This algorithm does not include the vertices of C in the recursive call (unlike the previous one). Also, the previous version of the algorithm made exactly two recursive calls at each step while this algorithm does one recursive call per connected component. Because this algorithm recurses on more than two subgraphs at each level it does not, in general, result in the same bounds for fill-in and operation count. However, the algorithm does give $O(n \log n)$ fill-in and $O(n^{\frac{3}{2}})$ total operation count for planar graphs, finite element graphs, graphs of bounded genus and graphs of bounded degree with $\sqrt{n}$ separators [GT87]. The constants in the fill bounds are smaller than in the previous version. For other classes of graphs with $\sqrt{n}$-separator theorems it may perform even better [Gil80].

In summary, Alan George's nested dissection algorithm solves a system of linear equations defined on an $n = k * k$ square grid. The generalized nested dissection algorithm, as its name suggests, is a generalization of this method to any system of equations defined on a planar or almost-planar graphs. Gilbert's algorithm as explained earlier is a minor modification of the generalized nested dissection algorithm.

3.3 Separator trees

The nested dissection algorithms are based on finding separators. The recursion of these algorithms suggests a natural decomposition of graphs in terms of their separators. At the highest level is a separator that divides the graph into components. These components themselves have separators, and so on. At the lowest levels are components that may not be divided any further (possibly singleton vertex sets). This decomposition can be described in terms of a structure called a separator tree. A separator tree for a graph is shown in Figure 11.

A separator tree for a graph G, hence, is a tree whose internal nodes are separators and whose leaves are the components of the graph G that may not be divided any further. Hence each node in the separator tree is a subgraph of the original graph G and may contain many vertices of G. In the original generalized nested dissection algorithm the separator trees are binary trees (2-ary). For Gilbert's modified algorithm the separator trees are k-ary ($k \geq 2$) while those in Alan George's method are 4-ary. The root of a separator tree is at level 0. The level of any node in the tree is the length of the path from the root to that node.

Lemma 3.3 *Let G = (V,E,α) ba an ordered graph. Then (v,w) is an edge of $G^\star_\alpha$ (defined earlier) if and only if there exists a path μ = [v=v_1, $v_2, \ldots, v_{k+1}$=w] in G_α such that*

$$\alpha^{-1}(v_i) < min(\alpha^{-1}(v), \alpha^{-1}(w)) \textit{ for } 2 \leq i \leq k$$

This lemma states that an edge (v, w) fills in if and only if there is a path from v to w containing only vertices deleted before either v or w. The lemma may be used to calculate bounds on fill-in due to nested dissection. Consider a node of the separator tree C, and its subtrees $C_1, C_2, \ldots, C_n$. There are no paths between C_i and C_j initially for $i \neq j$. The elements of C are given higher elimination numbers than those in C_i and C_j. Hence, there can not be a fill-in edge between any member of C_i and C_j. Thus, the separator tree shows that the only possible fill-in that may occur is along the edges of the tree, or between the vertices of an individual node of the tree. This fact may be used to calculate bounds on the total amount of fill-in using a nested dissection ordering algorithm for some classes of graphs.

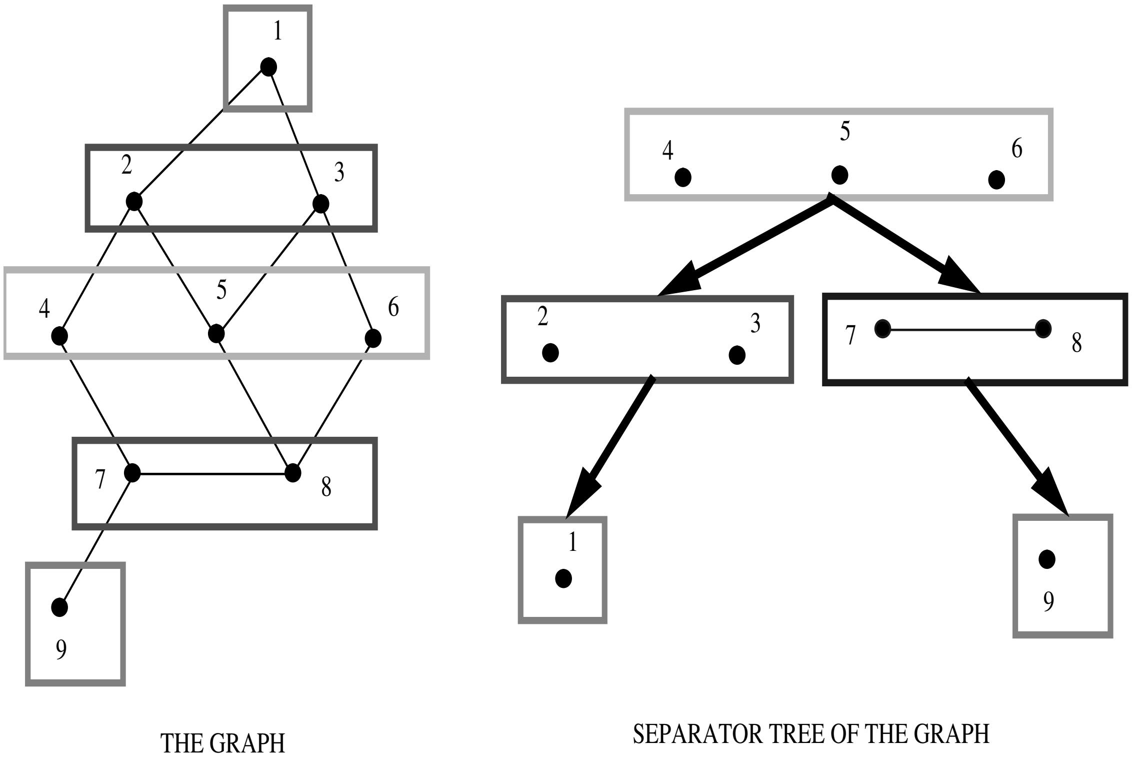


Figure 11: A graph and its family of separators

3.4 A bound on the fill for Gilbert's algorithm

In this section we will prove that Gilbert's algorithm causes $O(n \log n)$ fill in a planar graph. We actually prove this bound for a class of graphs that satisfies the $\sqrt{n}$-separator theorem and is closed under subgraph and contraction.

Lemma 3.4 ([GT87]) *Let S be a class of graphs that satisfies the $\sqrt{n}$-separator theorem with constants $\alpha < 1$ and $\beta > 0$ and is closed under subgraph and contraction. Suppose no n vertex graph in S has more than $\delta n + c$ edges. When Gilbert's nested dissection algorithm is applied to a graph in S with $n > n_0$ vertices, the number of fill edges with at least one endpoint in the top level separator C (the root of the separator tree) is $O(\delta n)$.*

Proof:

We shall refer to the nodes of the separator tree for G as nodes and to the vertices of graph G as vertices. Hence a node in the separator tree can have several graph vertices in it. Let $\mathcal{N}$ be the set of nodes of the separator tree for G, and let $\mathcal{N}_k$ be the set of nodes on level k of the tree. Thus, $\mathcal{N}_\prime = \{\mathcal{C}\}$ and

$$\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_1 \cup \ldots$$

For any given node N, let s_N be the number of vertices in N.

Consider level k of the separator tree. We will count fill to the root of the separator tree (say C) from nodes of the tree at level k. Every subtree rooted at level k is connected, since in Gilbert's algorithm every separator splits up the graph into a number of connected components. Contract each of these subtrees into a single vertex. Remove all the vertices of this graph except contracted vertices and vertices in C. Throw out edges between vertices in C. Let the resulting graph be G_k. Since G_k is obtained from G by contraction and removal of vertices and edges, G_k is in S and hence has at most $\delta|G_k| + c$ edges.

From the discussions in section 3.3, it is clear that there will be fill to a vertex v in C from a level k node N only if there is an edge in G_k from v to a contracted vertex corresponding to N. Each such edge accounts for at most one fill edge from each vertex of G in N, or s_N fill edges in all. Let f_k be the size of fill to C from level k nodes and e_N be the degree in G_k of the contracted vertex corresponding to node N. Hence,

$$f_k \leq \sum_{N \in \mathcal{N}_k} e_N s_N$$

Let $\mathcal{M}_k$ be the set of level k nodes with degree greater than δ in the contracted graph. Then,

$$\begin{aligned} f_k &\leq \sum_{N \in \mathcal{N}_k} \delta s_N + \sum_{\mathcal{M}_k} (e_N - \delta) s_N \\ &\leq \delta \sum_{N \in \mathcal{N}_k} s_N + \hat{s_k} \sum_{\mathcal{M}_k} (e_N - \delta) \; where \; \hat{s_k} = \max_{N \in \mathcal{M}_k} s_N \end{aligned} \tag{2}$$

Consider the subgraph of G_k that is induced by the vertices of C and the contracted vertices of $\mathcal{M}_{||}$. By the $\sqrt{n}$-separator theorem, it has at most $\beta n^{\frac{1}{2}} + |\mathcal{M}_k|$ vertices. The subgraph is in S so

$$\begin{aligned} \sum_{\mathcal{M}_k} e_N &< \delta(\beta n^{\frac{1}{2}} + |\mathcal{M}_k| + c \\ \sum_{\mathcal{M}_k} (e_N - \delta) &< \delta \beta n^{\frac{1}{2}} + c \end{aligned} \tag{3}$$

Equations (2) and (3) imply

$$f_k < \delta \sum_{N \in \mathcal{N}_k} s_N + (\delta \beta n^{\frac{1}{2}} + c)\hat{s_k}$$

Hence the total fill to C is

$$\sum_{k \geq 0} f_k < f_k < \left(\begin{array}{c} |C| \\ 2 \end{array} \right) + \delta \sum_{N \in \mathcal{N}} s_N + (\delta \beta n^{\frac{1}{2}} + c) \sum_{k > 0} \hat{s_k}$$

Now

$$\sum_{N \in \mathcal{N}} s_N = n \ \ and \ \ \left(\begin{array}{c} |C| \\ 2 \end{array} \right) < \beta^2 \frac{n}{2}$$

Also, it is easy to show that

$$\sum_{k>0} \hat{s_k} = O(n^{\frac{1}{2}})$$

Hence the total fill is $O(\delta n)$.

The fill when eliminating G is the union over every internal separator tree node of the fill edges whose higher-numbered vertex is in that node plus the fill edges within the external nodes (leaves) of the tree. A fill edge whose higher numbered endpoint is in a given internal node has its other endpoint in a descendant of that node. Thus if a given internal node is the root of a subtree containing m vertices then by the lemma just proved the number of fill edges with higher numbered endpoints in that node is $O(\delta m)$. If we sum this over all the internal nodes of the separator tree we get $O(\delta n \log n)$. The fill within an external node is atmost $\left(\begin{array}{c} n_0 \\ 2 \end{array} \right)$ edges, for a total over the whole graph of $O(n)$ edges. Thus the bound for fill for the entire graph is $O(\delta n \log n)$.

Now a planar graph with n vertices has at most $3n - 6$ edges. Planar graphs are closed under contraction because any edge in an embedding of the graph in a plane can be shrunk without disturbing the embedding. Planar graphs are also closed under subgraph. Hence, by the above analysis the fill that occurs in a planar graph due to Gilbert's algorithm is $O(n \log n)$. This analysis has been taken from [GT87].

3.5 A bound on operation count for Gilbert's algorithm

It can be shown that in a graph G, eliminating vertex v takes arithmetic operations propotional to the square of the degree of v ([GL81]). Let G^* be a perfect elimination graph corresponding to G. G^* contains not only the edges in G, but also the fill edges produced as a result of eliminating vertices in the order obtained by the application of the nested dissection algorithm to G. We will make G^* a directed graph by orienting each edge in G^* from the endpoint with lower ordering number to the endpoint with the higher one. Let the out-degree of v be $d(v)$. Then the cost of eliminating v is $O(d(v)^2)$. Hence, the operation count for the entire elimination is $O(\sum_v d(v)^2)$.

Let $\mathcal{N}$ be the set of nodes of the separator tree of G. Let $\mathcal{N}_k$ be the set of nodes on level k of the tree. Let

$$p_k = \sum_{v \in N \in \mathcal{N}_k} d(v)^2 \tag{4}$$

be the sum over all vertices of the square of the out-degree.

Now, every subtree rooted at level k is connected. Let G_k be the graph obtained by contracting each subtree into a single vertex and deleting all edges in G that are not incident on contracted vertices. Let v be a vertex of G in node N on level k of the separator tree, and let (v, w) be an edge of G^*. Now, the edges of G^* are directed edges from the lower-numbered endpoint to the higher-numbered endpoint. Also, because lower numbered vertices are at the same or higher level than higher numbered vertices in the separator tree, either v and w are both in node N, or there is an edge in G_k joining w and the contracted vertex corresponding to N. If s_N is the number of vertices in node N and e_N is the number of edges incident on contracted vertex

N in G_k, then $d(v)$ is at most $s_N + e_N$, so

$$p_k \leq \sum_{N \in \mathcal{N}_k} s_N(s_N + e_N)^2 \tag{5}$$

Lemma 3.5 ([GT87]) *Let G be a planar bipartite graph with n vertices (and hence at most $2n-4$ edges). There is a function ϕ from the edges of G to the vertices of G such that for all edges e, $\phi(e)$ is an endpoint of e; and for all vertices v, $\phi(e) = v$ for at most two different edges e.*

G_k is planar and bipartite. Hence by the lemma stated above we can associate each edge of G_k with one of its endpoints in such a way that at most two edges are associated with each vertex. Gilbert calls the edges associated with contracted vertices *red* vertices and those associated with vertices of G on levels 0 through $k-1$ of the separator tree *blue* edges. Of the e_N edges incident on contracted vertex N, let r_N be red and b_N be blue. By the lemma at most two edges are associated with N in G_k. So, $r_N \leq 2$ and $e_N = r_N + b_N$. So Equation 5 becomes

$$p_k \leq \sum_{N \in \mathcal{N}_k} s_N(s_N + r_N + b_N)^2 \tag{6}$$

The following mathematical inequality if well know: If a, b and c are real numbers then

$$(a+b+c)^2 \leq 3(a^2 + b^2 + c^2)$$

So,

$$\begin{aligned} p_k &\leq \sum_{N \in \mathcal{N}_k} s_N(3(s_N^2 + r_N^2 + b_N^2)) \\ &\leq 3\sum_{N \in \mathcal{N}_k} s_N^3 + 3\sum_{N \in \mathcal{N}_k} s_N r_N^2 + 3\sum_{N \in \mathcal{N}_k} s_N b_N^2 \\ &\leq 3\sum_{N \in \mathcal{N}_k} s_N^3 + 12\sum_{N \in \mathcal{N}_k} s_N + 3\hat{s}_k \sum_{N \in \mathcal{N}_k} b_N^2 \; where \; \hat{s}_k = \max_{N \in \mathcal{N}_k} s_N \end{aligned} \tag{7}$$

The bound on the first two terms of Equation 7 is easy to calculate. We examine the bound to

$$\sum_{N \in \mathcal{N}_k} b_N^2 \tag{8}$$

Consider some node M on level $r < k$ of the separator tree. The vertices in M are vertices of G_k. Each vertex has at most two blue edges incident on it (since G_k is planar and bipartite and by lemma 3.5). Since G_k has only those edges of G^* that are incident on the contracted vertices at level k, the other endpoints of the blue edges mentioned earlier are contracted vertices. Now, the blue edges out of M may be incident on different contracted vertices. But by examining Equation 8 it is clear that if all the blue edges out of M are incident on the same contracted vertex then the sum will be larger. Hence, we can assume that all blue edges coming from the vertices in the same node go to the same contracted vertex.

Now let N be a contracted vertex. Blue edges incident on N may come from many different levels. Let M be the node closest to the root such that a blue edge (v, N) exists for $v \in M$. Then, all the blue edges incident on N come from nodes on the tree path from M to N. If the number of vertices of G in the subtree rooted at M is n_M, the number of vertices of G on this tree path is at most

$$\beta n_M^{1/2} + \beta(\alpha n_M)^{1/2} + \beta(\alpha^2 n_M)^{1/2} + \ldots = O(n_M^{1/2})$$

So, b_N, the number of edges incident on N, is also $O(n_M^{1/2})$. Hence,

$$\sum_{N \in \mathcal{N}_k} b_N^2 \leq \sum_{M \in \mathcal{N}_r, 0 \leq r < k} cn_M \quad \textit{for some } c > 0 \tag{9}$$

Hence,

$$\sum_{N \in \mathcal{N}_k} b_N^2 \leq \sum_{0 \leq r < k} \sum_{M \in \mathcal{N}_r} cn_M \tag{10}$$

The subtrees rooted at level r are disjoint. So the inner sum is at most cn. Therefore the whole sum is ckn.

$$\sum_{N \in \mathcal{N}_k} b_N^2 \leq ckn \tag{11}$$

Substituting Equation 11 in Equation 7 and summing over all levels k yields

$$\sum_{v \in G^*} d(v)^2 \leq \sum_{N \in \mathcal{N}} s_N^3 + 12 \sum_{N \in \mathcal{N}} s_N + 3\hat{s_k} \sum_{N \in \mathcal{N}_k} ckn \tag{12}$$

Now $s_N \leq \beta n_N^{1/2}$ and $\hat{s_k} \leq \alpha^{k/2} \beta n^{1/2}$, this is at most

$$3\beta^3 \sum_{N \in \mathcal{N}} n_N^{3/2} + 12\beta \sum_{N \in \mathcal{N}} n_N^{1/2} + 3\beta c n^{3/2} \sum_{k \geq 0} k\alpha^{k/2}$$

The first sum is $O(n^{3/2})$ and the second is $O(n)$. The third sum converges to a constant, so the entire expression is $O(n^{3/2})$.

3.6 Euclidean norm and fill-in

The Euclidean norm of a graph G and its relation to the fill-in that gaussian elimination may cause is discussed in this section. Without loss of generality, assume that $G = G(M)$ is an embedded triangulated planar graph. In such graphs, the separators must be cycles.

Definition 3.6 *A simple cycle C is a simple cycle separator of G if the vertices interior to C are less than or equal to $\frac{2}{3}n$ in number and those exterior to C are also bounded by the same fraction.*

Lemma 3.7 ([Mil86]) *If G is a triangulated planar graph then there exists a simple cycle separator of size $\sqrt{8n}$ (for $(1/3, 2/3)-$separator).*

For the sake of simplicity, we will take $\frac{1}{2}n$ to be the bounding figure instead of $\frac{2}{3}n$.

Definition 3.8 *The element graph corresponding to G, $El(G) = (V, E')$ where*
E' = $\{(v, w) |$ v and w share a face in G$\}$

Example: For a triangulated graph G, $El(G)$ = G. If G is a simple cycle then $El(G)$ = $K_{|G|}$. Recall that pivoting caused cliques to form. So the element graph gives an idea of the amount of fill-in.

Given the matrix M, we will start with $G(M)$ and triangulate it. We will rip out vertices from $G(M)$ and replace them by a clique whose size is determined by the face created due to removal of the respective vertices. We will argue that the amount of fill-in created as a result of the removal of the vertices is bounded by the number of edges in the element graph corresponding to G. Thus, Total fill-in $\leq$ number of edges in the element graph of G. In the following discussion we will denote the number of edges in the element graph corresponding to G by Edges($El(G)$).

Definition 3.9 *The Euclidean Norm [GM90] of a graph G,* $||G||$ = $\sqrt{\sum d_i^2}$, *where* d_i *is the size of the* i^{th} *face.*

Note that the size of a face is the number of vertices or edges in that face.

Lemma 3.10 *Edges(*$El(G)$*) =* $O(||G||^2)$.

Proof: Let the i^{th} face have size d_i. There are two kinds of edges in an element graph corresponding to a graph G - edges already in G and cross face edges that are added. In the element graph each vertex in the i^{th} face is connected to $(d_i - 3)$ other vertices in the face by cross face edges (no edges will be added for the two adjacent vertices and the vertex itself). But since the edges are not directed we have to divide the total by a factor of 2. This gives us the first term in the equation below. The second term comes from the edges forming the boundary of the face. Here too to compensate for the double counting of these edges in adjacent faces we divide by a factor of 2 giving us

$$Edges(El(G)) = \sum(\frac{d_i(d_i - 3)}{2} + \frac{d_i}{2})$$

$$Edges(El(G)) = \sum(\frac{d_i^2 - 2d_i}{2})$$

$$Edges(El(G)) = \frac{||G||^2}{2} - \sum d_i$$

$$Edges(El(G)) = O(||G||^2)$$

So if we can show that $||G||$ is small, then we can conclude that fill-in will be small too.

3.7 Elimination ordering algorithms as tree traversals

The separator tree can be used as a framework for describing all the elimination ordering algorithms for sequential nested dissection. Alan George's nested dissection algorithm numbers the vertices in a separator tree in the order obtained by a *reverse level order* traversal of the separator tree. This ordering is the one obtained by numbering the vertices at the highest level in the graph and then moving to vertices at lower levels. The Generalized Nested Dissection algorithms use a postorder traversal of the separator tree to obtain the elimination order. Note that in Gilbert's modified version the separator tree may not be a binary tree. In conclusion, all the elimination ordering algorithms are different forms of tree traversals of the separator tree. In fact, any tree traversal algorithm that visits a vertex before any of the vertex's parents, when applied on a separator tree would result in an ordering that would produce a fill-in within (section 2.2) the required bound. Hence, Alan George's mesh nested dissection algorithm can also be described as a recursive algorithm as follows:

Assume that k is one less than a power of two.

- Remove row $(k+1)/2$ and column $(k+1)/2$. Give the highest numbers to these $2k-1$ vertices (i.e., these vertices will be eliminated last).
- There are now four components of the original graph. If their sizes are greater than one, recursively number the components. Otherwise number the four vertices in the range specified.

Another way of looking at the elimination ordering algorithms is as follows: Let *rake* [GMT88] be an operation that removes all leaves from a tree. The elimination ordering can be obtained by iteratively *raking* the separator tree. If T is the separator tree the following algorithm defines elimination ordering by the sequential algorithms discussed earlier. Let $Leaves(T)$ denote the set of leaves of the tree T, obtained as a result of the rake operation and $T^{'}$, the resulting tree after the operation.

- Rake T.
- If N denotes the largest elimination ordering number of the nodes numbered so far, then number the vertices in $Leaves(T)$ from $(N+1)$ to ($N+$ cardinality of the set of leaves).
- If T' is not empty then Repeat the above steps with $T = T'$.

Note that though *rake* is an operation usually discussed in a parallel context, here it is to be looked upon as a sequential operation with the same end result as in the parallel case i.e., removing all the leaves from the tree.

Thus in general all the sequential algorithms for determining the elimination ordering of a graph G can be described by the following general algorithm:

1. Generate the tree of separators for G.
2. Perform a tree traversal on the separator tree to order the vertices. This traversal must visit a node before any of its parents.

4 Parallel Nested Dissection

4.1 The basic parallel algorithm

The Parallel Algorithm devised by Pan and Reif [PR85a] [PR85b] is discussed here in graph theoretic terms.

Once again, we are trying to solve the system $Mx = b$, where M is a symmetric positive definite $n \times n$ matrix. The Pan-Reif parallel algorithm is based on computing a special recursive factorization of M. Assume that $G(M)$ belongs to a class of graphs that satisfy the $s(n)$-separator theorem for constants α and $\beta = 1$.

Definition 4.1 *A recursive $s(n)$-factorization of a matrix M with respect to α, $0 < \alpha < 1$, is a sequence of matrices $M_0, M_1, \ldots, M_d$ such that $M_0 = PMP^T, P$ is an $n \times n$ permutation matrix and for $h = 0, 1, \ldots, d-1$*

$$M_h = \begin{pmatrix} A_h & B_h^T \\ B_h & C_h \end{pmatrix}, C_h = M_{h+1} + B_h A_h^{-1} B_h^T$$

and A_h is a block-diagonal matrix consisting of square blocks of sizes at most $s(\alpha^{d-h}n) \times s(\alpha^{d-h}n)$ where $n_0 \geq \alpha^d n$, n_0 is a constant.

Now, given the recursive factorization of M, $M^{-1}b$ can be computed recursively. This comes from the following observation. By the definition of recursive factorization

$$M_h = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ C_h A_h^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} A_h & \mathbf{0} \\ \mathbf{0} & M_{h+1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & A_h^{-1} B_h^T \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

Hence ,

$$M_h^{-1} = \begin{pmatrix} \mathbf{I} & -A_h^{-1} B_h^T \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} A_h^{-1} & \mathbf{0} \\ \mathbf{0} & M_{h+1}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -A_h^{-1} B_h^T & \mathbf{I} \end{pmatrix}$$

The Algorithm

Let us assume that we are given a family of separators for $G(M)$. We will construct a separator tree. Let the separator tree have k levels. Let V_j be the set of vertices (corresponding to variables in M) in the nodes of the separator tree at any level j. The following procedure computes the inverse of a matrix N recursively.

`compute_inverse`(matrix N, level j)

Begin

If (j equals 0) return N^{-1}

Else

Begin

write N so that the variables of V_k are in the top right hand corner of N i.e.,

$N = \begin{pmatrix} A & B^T \\ B & C \end{pmatrix}$, where A are variables from V_k.

$D = C - BA^{-1}B^T$

D^{-1} = `compute_inverse`$(D, k-1)$

inverse = $\begin{pmatrix} I & -A^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & D^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -BA^{-1} & I \end{pmatrix}$

return inverse

End

End

The procedure is initially called with $N = M$ (matrix) and $j = k$ (level).

The following lemma is taken from [PR85a].

Lemma 4.2 ([PR85a]) *Given an $n \times n$ positive definite symmetric matrix M such that $G(M)$ satisfies the $s(n)$-separator theorem and $s(n)$ is of the form n^σ for a constant σ, then the Pan-Reif algorithm can compute* *the special recursive factorization of M in $O((log n)(log(s(n)))^2)$ time using |E| + $M(s(n))$ processors. Then, given this recursive factorization, the solution of $Mx = b$ for any given b requires only O(log n log s(n)) time and |E| + $(s(n))^2$ processors.*

4.2 An example with lots of fill-in

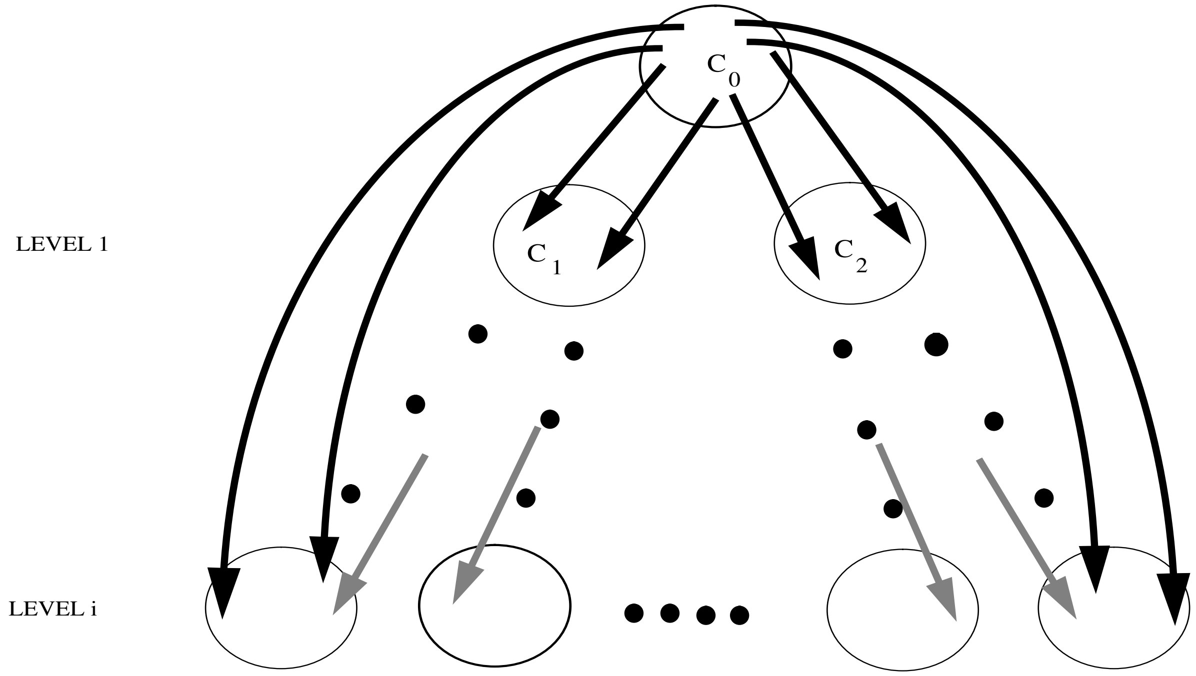
In this section we consider an example of nested dissection that results in a lot of fill and find an upper bound on the amount of fill that results. Let us consider the figure below which shows a family of separators. In this tree each vertex at a particular level is connected to all its ancestors by edges.

The number of vertices in the separator at the root is $\sqrt{n}$. The separator splits the graph into two subgraphs of size $\frac{n}{2}$ each, so separator sets C_1 and C_2 are each of size $\sqrt{\frac{n}{2}}$. Thus, in general the size of a separator at level i is $\sqrt{\frac{n}{2^i}}$ and there are 2^i such separators. Hence we can write the vertex count $|V|$ as,

$|V| = \sqrt{n} + 2\sqrt{\frac{n}{2}} + 4\sqrt{\frac{n}{4}} + \ldots + 2^i\sqrt{\frac{n}{2^i}}$.

and the edge count $|E|$ can be written as,

$|E| = 2\sqrt{\frac{n}{2}}\sqrt{n} + 4(\sqrt{\frac{n}{4}}\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{4}}\sqrt{n}) + \ldots + 2^i \sum_{k=i-1}^{0} \sqrt{\frac{n}{2^i}}\sqrt{\frac{n}{2^k}}$.



AN EXAMPLE WITH LOTS OF FILL-IN

Now that we have a fair idea about the graph, let us calculate the fill-in. Consider the blocks of vertices at level i. At level i there are 2^i blocks of size $\sqrt{\frac{n}{2^i}}$ each. We will eliminate these blocks of vertices first. Recall that for each block (call the corresponding matrix A) we have to find the inverse (A^{-1}). For the sake of convenience let

$$p = \sqrt{\frac{n}{2^i}}$$

The calculation of inverse takes $O(p^3)$ time for a matrix of size p. Hence the total time to find inverses is $2^i p^3$, since there are 2^i such matrices. We can simplify this expression to $n^{\frac{3}{2}}/\sqrt{2^i}$. Now we need to calculate $\overline{D} = D - CA^{-1}B$. We calculate $\overline{C} = CA^{-1}$ first and then $\overline{C}B$ to get $CA^{-1}B$. Recall that the taking the inverse of a matrix is equivalent to finding the Transitive Closure of the corresponding graph.

So in graph theoretic terms finding CA^{-1} is equivalent to finding all possible paths from ancestral vertices to a vertex in a block of separators (A) at the ith level and then onto another vertex in the same block. There are

$$O(\sqrt{n}\sqrt{\frac{n}{2^i}}\sqrt{\frac{n}{2^i}}2^i) = O(n^{\frac{3}{2}})$$

such paths. Note that this term is obtained by taking the dominant term from a geometric series obtained by considering all possible paths. The dominant term comes from finding all possible paths from C_0 ($\sqrt{n}$ vertices) to a vertex in a separator at level i ($\sqrt{\frac{n}{2^i}}$) and then onto another vertex in the same separator.

Similarily, evaluating $\overline{C}B$ is finding all paths starting at levels less than i to a block of vertices at level i and then back to the original level. There are

$$O(\sqrt{n}\sqrt{\frac{n}{2^i}}\sqrt{n}\sqrt{2^i})$$

such paths. This is atmost $(O(n^2))$ since $\sqrt{2^i} \leq n$.

4.3 A comparison with the sequential algorithm

Both the algorithms, parallel and sequential are based on finding separators. However, the process of factorization of the given matrix is different. The parallel case uses a recursive factorization as described in section 4.1. In the case of the Pan-Reif algorithm, the length of the factorization is $\Omega(logn)$. Sequential algorithms use the LDL^T factorization described in section 2.1 which is $O(n)$ in length. In the sequential case elimination ordering was described in section 2.4 using separator trees and the rake operation. In the parallel version the idea is similar. A separator tree can be drawn again and it will be identical to that obtained in the sequential case. The rake operation can be used again to understand the elimination ordering of the vertices. This ordering is similar to that in the sequential case, except that after every rake operation, all the vertices in $Leaves(T)$ are eliminated in parallel (the vertices in matrix A_0 mentioned earlier). This means that all the vertices in $Leaves(T)$ will get the same elimination ordering number.

The algorithm is given below for the sake of completeness:

level = height of separator tree

1. Rake T. Let T' be the resulting tree.
2. Number the vertices in $Leaves(T)$ by the same number i.e. (height - level). level = level -1
3. If T' is not empty then Repeat the above steps with $T = T'$.

The parallel nested dissection algorithm eliminates all vertices with the same level simultaneously as was indicated in section 3.1. and in increasing elimination ordering numbers.

Thus the algorithms for determining the elimination ordering of a graph G can be described by the following algorithm:

1. Generate the tree of separators for G.

2. Label the vertices as explained earlier (using the rake operation).

3. Eliminate the vertices in the ordering obtained by step 2.

5 Conclusion

In this survey, we have tried to explain nested dissection in graph theoretic terms. A common framework of separator trees has been used to compare some of the popular nested dissection algorithms. The sequential nested dissection algorithms have been shown to involve different kinds of tree traversals of the separator tree for the given graph. Bounds for the fill-in and operation count for Gilbert's nested dissection algorithm have been calculated. The parallel algorithm is shown to be a modification of the sequential one and has been explained using the framework of separator tree.

6 Acknowledgements

This survey arose out of the class notes on Nested Dissection from Gary Miller's Parallel Algorithms course. The details in the survey were filled in from the referenced papers and also Tom Sheffler's Master's Thesis [She87]. In fact sections 2.2, 2.4 and 3.2 are taken from his thesis. The lecture notes on advanced parallel and VLSI computation held at MIT in the spring of 1987 was also used as a reference [LLM+88]. I would also like to acknowledge CRK Prasad, Stephen Guattery, Dafna Talmor and Jeff Jackson who not only reviewed various versions of these notes but also helped me understand some of the papers. Finally a word of thanks to Tom Sheffler whose discussions helped me start out on this topic.

References

[Dji81] H. N. Djidjev. A separator theorem. *Compt. End Acad. Bulg. Sci.*, 34(5):643–645, 1981.

[Geo73] A. George. Nested dissection of a regular finite element mesh. *SIAM J. on Numerical Analysis*, 10:345–363, 1973.

[Gil80] J. R. Gilbert. *Graph Separator Theorems and Sparse Gaussian Elimination*. PhD thesis, Stanford University, Department of Computer Science, 1980.

[GJ79] M. R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, New York, NY, 1979.

[GL81] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[GM90] Hillel Gazit and Gary L. Miller. Planar separators and the Euclidean norm. In *SIGAL International Symposium on Algorithms*, Tokyo, August 1990. Information Processing Society of Japan, Springer-Verlag. to appear.

[GMT88] H. Gazit, G. L. Miller, and S-H Teng. Optimal tree contraction in an EREW model. In S. K. Tewksbury, B. W. Dickinson, and S. C. Schwartz, editors, *Concurrent Computations: Algorithms, Architecture and Technology*, pages 139–156, New York, 1988. Plenum Press. Princeton Workshop on Algorithms, Architecture and Technology Issues for Models of Concurrent Computation.

[GT87] J. R. Gilbert and R. E. Tarjan. The analysis of a nested disection algorithm. *Numerische Mathematik*, 50:377–404, 1987.

[Joh87] Claes P. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambride University Press, The Pitt Building, Trumpington Street, Cambridge CB2 1RP, 1987.

[LLM+88] Tom Leighton, Charles E. Leiserson, Bruce Maggs, Serge Plotkin, and Joel Wein. Advanced parallel and vlsi computation lecture notes. *Research Seminar Series, LCS, Massachusetts Institute of Technology*, (MIT/LCS/RSS 2), March 1988.

[LRT79] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[LT80] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. on Computing*, 9:615–627, 1980.

[Mil86] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, June 1986. invited publication.

[Par61] S. V. Parter. The use of linear graphs in gaussian elimination. *SIAM Rev.*, 3:119–130, 1961.

[PR85a] Victor Pan and John Reif. Efficient parallel solution of linear systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 143–152, Providence,RI, May 1985. ACM.

[PR85b] Victor Pan and John H. Reif. Extension of parallel nested dissection algorithm to the path algebra problems. Technical Report TR-85-9, Computer Science Department, State University of New York at Albany, New York, 1985.

[Ros72] D. J. Rose. *A Graph Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations*, pages 184–218. Academic Press, 1972.

[She87] Thomas J. Sheffler. A graph separator theorem and its application to gaussian elimination to optimize boolean expressions for parallel evaluation. *Carnegie Mellon University, School of Comp. Sc. Technical Report*, (CMU-CS-87-123), February 1987.