Approximating Fill in Solving Sparse Linear Systems

Ajit Agrawal
Philip Klein
R. Ravi

Department of Computer Science
Brown University
Providence, Rhode Island 02912

Technical Report No. CS-91-19
March 1991

Approximating fill in solving sparse linear systems

Ajit Agrawal[†] Philip Klein[1 2] R. Ravi[†]

[1]Research supported by an NSF grant CCR-9012357

[2]Brown University, Providence, RI 02912. Reseach supported by the National Science Foundation under NSF grant CDA 8722809, by the Office of Naval and the Defense Advanced Research Projects Agency under contract N00014-83-K-0146, and ARPA Order No. 6320, Amendment 1.

Abstract

In this paper, we present the first approximation algorithm for minimizing fill in solving a symmetric positive-definite sparse linear system of equations. Moreover, the elimination ordering of variables output by the algorithm also approximately minimizes two other important quantities: the total number of multiplications, and the total time taken on a parallel machine to solve the problem using gaussian elimination. For a constant number of nonzero elements in each row and column, the performance guarantee is polylogarithmic in the number of variables. The bound is weaker in other cases.

The problem of minimizing fill is directly related to the problem of minimum graph chordalization for which we give an approximation algorithm here. The algorithm is based on a method of Leighton and Rao for finding approximate balanced node separators in undirected graphs.

1 Introduction

It has been a longstanding open problem to find an approximately minimum *chordalization* of a graph, i.e. to find a set of edges whose addition to the input graph yields a chordal graph. This problem has several applications, of which the most well-known was discovered by Rose [12], Gaussian elimination for sparse linear systems. When Gaussian elimination is used to solve a sparse symmetric positive-definite linear system, the order in which variables are eliminated has a significant bearing on the complexity of the elimination process, both in terms of time and space. The reason is that each successive variable elimination typically results in "fill-in", new non-zero entries in the matrix. These new non-zero entries must be stored, and they enter into subsequent calculations, so in order to minimize storage and calculation time, it is desirable to minimize the fill-in. Heuristics such as the "minimum-degree" heuristic are commonly used in an attempt to do just that. Other heuristics can be found in [6].

Rose reformulated the notion of fill-in as a graph property. He showed that if we interpret the matrix as a graph—where a non-zero entry A_{ij} corresponds to the existence of an edge $\{i, j\}$—then the filled-in matrix corresponds to a *chordal* graph. He thereby characterized the matrices for which no fill-in is required—they are the matrices whose graphs are chordal. He also demonstrated that for any given matrix, minimizing the fill-in corresponds to adding a minimum number of edges to a graph to make it chordal.

The issue of minimizing fill-in is especially important in the solution of linear programs. When an interior-point method is used, the principal cost of an iteration is due to solving a symmetric positive-definite system of linear equations. The particular system changes at each iteration, but the graph associated with the system remains the same. Since it is the graph that determines what fill-in is achievable and how, it is worth spending considerable time finding a good chordalization of the graph in order to save time and storage at each iteration of the linear programming algorithm.[1]

Unfortunately, as Rose, Tarjan, and Lueker conjectured [13] and Yannakakis proved [15], finding a minimum chordalization is NP-complete. We are thus led to considering whether minimum chordalization can be approximated by a polynomial-time algorithm. Until now, no such approximation algorithms were known that provably achieved better-than-trivial performance.

Our new algorithm

In this paper, we report the discovery of such an approximation algorithm. The algorithm is based on a method of Leighton, and Rao [8] for finding an approximate sparsest cut in an undirected graph, on a modification of this method due to Leighton, Makedon, and Tragoudas [9], and on a separator theorem of Gilbert, Rose, and Edenbrandt [5]. The algorithm performs particularly well on graphs with small degree. Fortunately, linear systems arising in practice (e.g. from the finite-element method) often have graphs with small degree, because of constraints in the physical system being modelled.

Theorem 1.1 *For an input graph with n nodes and maximum degree k, the algorithm outputs a chordal graph such that the number of edges is within a factor of $O(\sqrt{k}\log^4 n)$ of optimal.*

We can also prove a performance bound when the degree is large, but the bound is weaker and somewhat complicated, and we postpone it until Section 3.

The value of *fill* in solving a sparse linear system equals the sum of the number of non-zero elements in the original matrix and the fill-in introduced by the elimination process. Fill characterizes the total space required to solve the sparse system of linear equations. It follows from the work of Rose that the value of minimum fill is related to the number of edges in the optimal chordal extension of the corresponding input graph. Theorem 1.1 thus implies that one can find an order of elimination that approximately minimizes the storage required for symmetric positive-definite matrices where each row and column has a small number of non-zero entries.

Our algorithm is similar to the generalized nested dissection of Lipton, Rose, and Tarjan [10]. Their algorithm works for any class of graphs with $O(f(n))$ separators, and relies on a recursive decomposition based on such separators. The performance of the algorithm is determined by the function $f(n)$. Since we do not assume the input graph has any special structure, we cannot rely on the existence of small separators. Gilbert [2] has shown that for every graph with degree at most k, there exists a nested dissection order that yields fill at most $O(k \log n)$ times optimum. We build on his important result. Gilbert's argument, however, is inherently non-constructive in that his choice of separators depends crucially on the optimally filled graph. Our analysis shows that it is sufficient to choose nearly optimal node separators in order to achieve near-optimal fill.

It turns out that our algorithm also approximately minimizes the total time required for the elimination process. Here we use the customary measure of time for this process, the number of multiplications.

Theorem 1.2 *The algorithm outputs an elimination ordering such that the number of multiplications is within a factor of $O(k \log^6 n)$ of optimal.*

[1] Note also that the cost of a floating-point operation, which depends on the precision required, typically exceeds the cost of a step in a chordalization algorithm.

Solving linear systems in parallel

Our algorithm is particularly well-suited to use in parallel solution of linear systems. In a typical parallel implementation of Gaussian elimination, each iteration consists of selecting some set of variables and eliminating them from the system. In order to eliminate all the variables in the set simultaneously, they must not interact—no two variables can occur in the same equation. The *height* of an elimination ordering is the number of iterations necessary; thus minimizing height corresponds to minimizing the parallel time required by this method. Moreover, the choice of which variables to eliminate in each step determines fill-in; it is desirable to keep the fill-in small, both to keep the storage small and to keep the number of processors small. Gilbert [3] has conjectured that there is an ordering that simultaneously minimizes height and approximately minimizes (to within a constant factor) the number of edges in the filled-in graph. Our analysis represents progress towards proving this conjecture.

Theorem 1.3 *For degree k graphs, there exists an ordering that simultaneously minimizes height to within a logarithmic factor and minimizes number of edges to within a factor of $O(\sqrt{k}\log^2 n)$.*

Theorem 1.3 follows from the analysis we use to prove Theorem 1.1 and from the analysis of Gilbert and Hafsteinsson [4]. They propose essentially the same algorithm as us to find elimination orderings that simultaneously minimize three important measures: tree width, front size, and elimination tree height. As part of their analysis, they show that the elimination tree height is within a logarithmic factor of the size of a minimum balanced node separator. Their algorithm differs from ours only in that it finds small node separators by first finding small edge separators. Consequently, their algorithm's performance guarantee for height depends linearly on the maximum degree of the input graph. As they observe, use of an approximation algorithm for node separators would yield a guarantee that was independent of degree. Since we use such an approximation algorithm, we achieve such a guarantee.

Theorem 1.4 *The algorithm outputs an elimination ordering whose height is within a factor of $O(\log^2 n)$ of optimal.*

2 Separators in Graphs

As a consequence of the approximate max-flow min-cut theorem of Leighton-Rao, one can find weighted edge- and node-separators in a graph. We list their results below.

Lemma 2.1 ([8, 9]) *For a graph (directed or undirected), there exists a polynomial algorithm to find a $\frac{1}{3}$-balanced edge-separator (node-separator) with cost within $O(\log n)$ factor of the optimal $\frac{1}{2}$-balanced edge-separator (node-separator).*

We use the above lemma for the approximation algorithm presented in the next section.

3 Minimum chordalization

In this section we give an approximation algorithm for minimum chordal graph completion of an input graph. For bounded degree graphs, the algorithm guarantees a polylog factor approximation to the number of edges in the optimal chordal graph. For unbounded degree graphs, the bound is weaker. This is the first known polynomial-time algorithm with a non-trivial performance guarantee.

As Rose has shown, and as discussed in the introduction, chordalization arises in solving a symmetric positive-definite system of linear equations. The order of elimination affects the storage, sequential time, and parallel time required to solve the system. Our algorithm outputs an ordering that approximately minimizes all these quantities simultaneously over all possible elimination orderings.

3.1 The Algorithm: Near-Optimal Generalized Nested Dissection

Given a graph $G(V, E)$, we define an elimination ordering α of its vertices. The graph G is then augmented to be the elimination graph G^* for the elimination ordering α [13]. G^* is the output chordal graph produced by the algorithm. The elimination graph is obtained as follows. At step i, the graph G_{i-1} is augmented to G_i such that all the higher numbered neighbors in G_{i-1} of the node numbered i (in the ordering α) form a clique. G_0 is set to the original graph G. The elimination graph corresponds to $G_{|V|}$.

Elimination Ordering Algorithm: Given a graph $G(V, E)$ with n nodes to be numbered in the range $[a, b]$, where $b = a + n - 1$, we proceed as follows. If $n = 1$, we number the node. Else, we find a balanced node separator X for G using Lemma 2.1. Let $|X| = s$. We number the vertices in the separator from $b - |X| + 1$ to b in any order. Let there be k connected subgraphs $A_1, \ldots, A_k$ of sizes $n_1, \ldots, n_k$ left on removing X from G. We recursively number the graph A_i in the range $[a+\sum_{j=1}^{i-1} n_j, a-1+\sum_{j=1}^{i} n_j]$ for each $i \in [1, k]$.

Separator Tree Representation: The elimination ordering of G can be related to the separator tree of the graph. The vertices in the separator X form the root of the tree with subtrees formed recursively for each of the pieces $A_1, \ldots, A_k$. The elimination numbering of the vertices is consistent with a postorder traversal of the tree nodes, with vertices in a tree node numbered in any order.

Defn: We shall refer to a node in the separator tree T as a *separator*, and to a node in the original graph G as a *vertex*. The vertices forming a separator are said to *belong* to the separator. Each vertex belongs to exactly one separator and we refer to this node as the *separator node* of the vertex. The *depth* of a separator is the distance of the separator node from the root of the tree. The depth of a vertex is the same as the depth of the separator it belongs to. The *subtree of a separator* is the subtree rooted at the separator in the separator tree. We say that a vertex u is an ancestor of v if u's separator node is either the same or is an ancestor of v's separator node in the separator tree.

We now state without proof a few lemmas that will be useful in subsequent arguments. All references to elimination orders or separator trees in Section 3 refer to the ones presented above.

Fact 3.1 *Every node induced subgraph of a chordal graph is also chordal.*

Theorem 3.1 ([5]) *Every chordal graph has a clique separator, and hence has a node separator of size at most $O(\sqrt{E})$, where E is the number of edges in the graph.*

Lemma 3.1 *For any vertex v, an edge (u, v) is in G^* only if for some edge $(z, v) \in G$, v is an ancestor of z, and u belongs to a separator node on the path from v's separator node to z's separator node. Note that this includes the case in which u and v belong to the same separator.*

Lemma 3.2 *The height of the separator tree is $O(\log n)$.*

3.2 Bounds on the number of edges

In this section we analyze the total number of edges in the resulting graph G^* for the cases of bounded and unbounded degree graphs.

Defn: We define the *weight of a separator* (w) to be the number of vertices belonging to the separator, and the *weight of a tree* (W) to be the sum of the weights of each of the separators in the tree. Edges with both their endpoints in vertices at the same depth and different depths will respectively be called *flat* edges and *jump* edges. The *cost of a vertex* (c) is the number of edges from the vertex to vertices at greater depth. Thus the total number of jump edges in G^* is the sum of the costs of the vertices. The *cost of a separator* is the sum of the costs of the vertices belonging to it.

Theorem 3.2 *The total number of flat edges in G^* at a given depth is $O\left(E_{opt}\log^2 n\right)$. Hence the total number of flat edges in G^* is $O\left(E_{opt}\log^3 n\right)$.*

Proof Sketch: By Lemma 3.1, the number of flat edges at a given depth are no more than those required to turn each separator into a clique. Let the subgraph induced by the subtree rooted at a separator X in the optimal chordal graph have $E_{opt,X}$ edges. By Fact 3.1, Theorem 3.1, and Lemma 2.1, $|X|$ is $\sqrt{E_{opt,X}}O(\log n)$. Hence the number of flat edges within X is $O(E_{opt,X}\log^2 n)$. Summing over node disjoint chordal graphs induced by the subtrees of each of the separators at the given depth, the claim follows. □

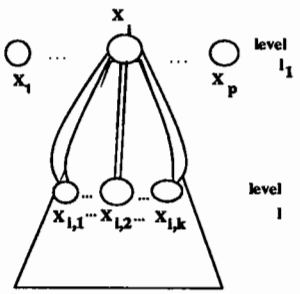

Figure 1: Counting the jump-edges in G^*- Bounded degree case.

Bounded degree graphs

Theorem 3.3 *The total number of edges in G^* is $O\left(\sqrt{k}E_{opt}\log^4 n\right)$, where k is the maximum degree of the graph.*

Proof: Theorem 3.2 shows that the total number of flat edges is within the bound. Hence it suffices to show that the total number of jump edges is $O\left(\sqrt{k}E_{opt}\log^4 n\right)$.

For a vertex u, consider its neighbors in G at greater depth. Consider the tree formed by the union of the paths from u's separator to separators containing such neighbors. Call it the *associated tree* of u. Since G has bounded degree k, this tree can have at most k separator leaves, and hence at most k separators at any depth. By Lemma 3.1, the cost of the vertex u is the weight of u's *associated subtree*.

Let us estimate the sum of the costs of all vertices at a given level ℓ_1 in the tree (see Figure 2). Suppose that level consists of separators $X_1, \ldots, X_p$. For $i = 1, \ldots, p$, consider the highest-cost vertex of X_i, and let T_i be the associated subtree for this vertex. For each level ℓ, let $W_\ell(T_i)$ be the weight of T_i due to vertices at level ℓ. Then the sum of the costs of vertices at level ℓ_1 is no more than the sum, over all levels ℓ greater than ℓ_1, of the value

$$\sum_{i=1}^{p} |X_i| \cdot W_\ell(T_i). \tag{1}$$

The weight of T_i at level ℓ is the sum of the sizes of at most k separators $X_{i,1}, \ldots, X_{i,k}$. Substituting into (1), we get

$$\sum_{i=1}^{p}\sum_{j=1}^{k} |X_i| \cdot |X_{i,j}| \leq \sqrt{\sum_{i=1}^{p} k|X_i|^2}\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{k} |X_{i,j}|^2} \tag{2}$$

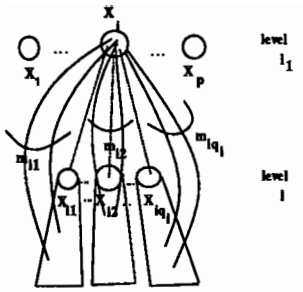

Figure 2: Counting the jump-edges in G^*- unbounded degree case.

where the inequality follows from the Cauchy-Schwartz inequality.

Since the X_i's are all disjoint, it follows from the proof of Theorem 3.2 that $\sqrt{\sum_i k|X_i|^2} = O(\sqrt{kE_{opt}}\log n)$. Similarly, $\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{k}|X_{i,j}|^2} = O(\sqrt{E_{opt}}\log n)$. Thus the right-hand side of (2) is $O\left(\sqrt{k}E_{opt}\log^2 n\right)$. Summing over all levels ℓ_1 and ℓ, we conclude that there are $O\left(\sqrt{k}E_{opt}\log^4 n\right)$ jump-edges. □

Unbounded degree graphs

For unbounded degree graphs, we have the the following result.

Theorem 3.4 *For an unbounded degree graph G with n vertices and m edges, the total number of edges in G^* is $O\left(E_{opt}{}^{\frac{3}{4}}\sqrt{m}\log^{3.5} n\right)$.*

Proof: As before, Theorem 3.2 bounds the total number of flat edges. We shall show that the total number of jump edges is no more than $O\left(E_{opt}{}^{\frac{3}{4}}\sqrt{m}\log^{3.5} n\right)$.

Let us first estimate the cost of all the vertices at a given level l_1. Let this level have p separators $X_1, \ldots, X_p$, and let T_i be the tree rooted at separator X_i. Consider another level l greater than l_1 (see fig. 2). We will estimate the number of edges between the vertices in X_i and the vertices in T_i at level l in the chordal graph G^*. Consider a separator $X_{i,j}$ belonging to T_i at level l, and let $T_{i,j}$ be the tree rooted at this separator. The vertices in $X_{i,j}$ have edges in G^* only to those vertices in X_i that have an edge in the original graph to a vertex in $T_{i,j}$. The number of such vertices in X_i is the minimum of $|X_i|$ and $m_{i,j}$, where $m_{i,j}$ is the total number of edges between vertices of X_i and vertices of $T_{i,j}$ in the original graph. Thus the total number of edges between X_i and $X_{i,j}$ in the chordal graph is at most $\min(|X_i|, m_{i,j}) \cdot |X_{i,j}|$, and hence at most $\sqrt{|X_i|\, m_{i,j}} \cdot |X_{i,j}|$. Summing over all the q_i separators belonging to T_i at level l, and again summing over all the p separators at level l_1, we get the total number of edges between vertices at level l_1 and vertices at level l. This value is given by

$$\sum_{i=1}^{p}\sum_{j=1}^{q_i}\sqrt{|X_i|\, m_{i,j}} \cdot |X_{i,j}| \tag{3}$$

Once again using Cauchy-Schwartz inequality, (3) is at most

$$\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q_i}|X_i|\, m_{i,j}} \cdot \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q_i}|X_{i,j}|^2} \tag{4}$$

Let m_i denote the sum $\sum_{j=1}^{q_i} m_{i,j}$. $\sum_{i=1}^{p}\sum_{j=1}^{q_i}|X_i|\, m_{i,j}$ then equals $\sum_{i=1}^{p}|X_i|\, m_i$. Using Cauchy-Schwartz inequality

$$\begin{aligned}\sum_{i=1}^{p}|X_i|\, m_i &\leq \sqrt{\sum_{i=1}^{p}|X_i|^2} \cdot \sqrt{\sum_{i=1}^{p} {m_i}^2} \\ &= O(\sqrt{E_{opt}}\log n \cdot m) \end{aligned} \tag{5}$$

Substituting (5) and $O\left(\sqrt{E_{opt}}\log n\right)$ for $\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q_i}|X_{i,j}|^2}$ into (4), we derive that the total number of jump edges between a given pair of levels is $O\left(E_{opt}{}^{\frac{3}{4}}\sqrt{m}\log^{1.5} n\right)$. Summing over $O\left(\log^2 n\right)$ choices of the pair of levels l_1 and l, we obtain the result. □

It should be noted that the above value is no more than $O(m^{\frac{1}{4}}\log^{3.5} n)$ factor of the optimal.

3.3 Bounds on the number of multiplications

For solving a symmetric sparse system of linear equations without pivoting, we view the matrix as the adjacency matrix of a graph G. We define the elimination numbering of the nodes in G as in Section 3.1. We show that for a matrix with the number of non-zero entries in any row (or column) not exceeding k, the number of multiplications performed using this elimination order is within a $O\left(k\log^6 n\right)$ multiplicative factor of that required by the optimal elimination ordering.

Before we prove the main result of this section, we give some lemmas that will be used later. A system of linear equations in n variables given by $Ax = b$ is considered dense if the matrix A has $\Theta(n^2)$ non-zero entries.

Fact 3.2 *The number of multiplications required to solve a dense system of equations in m variables is $\Omega\left(m^3\right)$.*

The following lemma is a consequence of fact 3.2.

Lemma 3.3 *For any chordal graph G^*, if m is the size of its clique separator, then $\Omega\left(m^3\right)$ is a lower bound on the number of multiplications required for any elimination ordering.*

Let M_{opt} be the optimal multiplication count for any elimination ordering of G. Then we have the following results.

Theorem 3.5 *Let a given level in the separator tree have p separators with weights $w_1 \ldots w_p$. Then $\Omega\left(\sum_{i=1}^{p} \frac{{w_i}^3}{\log^3 n}\right)$ is a lower bound on M_{opt}.*

Proof Sketch: The vertices in the subtrees of different separators at a level are disjoint. Hence the subgraphs induced by them in the optimal chordal graph are node disjoint. Each induced subgraph is also chordal and hence Lemma 3.3 provides a lower bound on the optimal multiplication count for each. By summing over all the subgraphs, we obtain the theorem. □

For any vertex with elimination number i, the vertex along with all its neighbors with number higher than i form a clique in G^*. We shall refer to this clique as the *associated clique* for the vertex, and denote it by C_v. The number of multiplications (M_v) required to eliminate a variable v, is the total number of edges in the clique C_v. Let M be the number of multiplications required for the elimination ordering defined by the algorithm. M is given by $\sum_v \sum_{e \in C_v} 1$, which is the same as $\sum_e \sum_{v: C_v \ni e} 1$. We can hence sum over the multiplication contribution for each edge. The *multiplication contribution* (henceforth referred to simply as *contribution*) for an edge (v, u) is the total number of vertices with depth no less than that of u or v, containing (v, u) in its associated clique.

Theorem 3.6 *The elimination ordering defined above yields a multiplication count of $O\left(kM_{opt}\log^6 n\right)$.*

Proof Sketch: We count the contributions from edges that go between any two levels of the tree. To count this contribution, we count the contribution due to all vertices at a depth no less than those of the two levels. Using techniques similar to those for proving Theorem 3.3 we can show that this contribution is no more than k times the sum of the cubes of the separator sizes at the three levels under consideration, which using Theorem 3.5 is $O\left(3kM_{opt}\log^3 n\right)$. The claim then follows. □

3.4 Bounds on the height of the elimination ordering

Minimizing the height of the elimination ordering minimizes the time required to solve the system of equations in parallel. Let h_{min} be the minimum elimination height for any elimination ordering of a given set of linear equations. We show that the elimination ordering proposed also minimizes height to within a polylog factor of h_{min}.

Fact 3.3 *To solve a dense system of equations in m variables, h_{min} is $\Omega(m)$.*

Lemma 3.4 *For any chordal graph G^*, if m is the size of its clique separator, then h_{min} for G^* is $\Omega(m)$.*

Theorem 3.7 *If w_{max} is the maximum weight of a separator in the separator tree of G, then h_{min} for G is $\Omega\left(\frac{w_{max}}{\log n}\right)$.*

Proof: Let X be the separator in the separator tree with the maximum weight, and let V_i be the set of vertices in the subtree of the separator X. If G^*_{opt} corresponds to the optimal chordal graph with minimum height over all elimination orders, then let the graph induced by V_i in G^*_{opt} be G_i. G_i has a clique separator of size $\Omega\left(\frac{w_{max}}{\log n}\right)$ by Theorem 3.1, and this is a lower bound on h_{min} by Lemma 3.4. □

Theorem 3.8 *The elimination ordering defined in Section 3.3 yields a height of $O\left(h_{min}\log^2 n\right)$.*

Proof: Consider all the separators at each level. One variable from each of the separators can be eliminated simultaneously as there are no direct edges between the variables. Hence the height for eliminating all the variables at a level is no more than w_{max}. Since the number of levels is $O(\log n)$, the claim follows. □

It should be mentioned that a parallel prefix operation may be required at each elimination step to update the coefficients in the matrix as a result of eliminating multiple variables simultaneously.

4 Final remarks

We presented an algorithm for approximating fill in solving a sparse linear system of equations. The algorithm performs well for graphs with small degree. We can show that the analysis of our algorithm is asymptotically tight. It remains an open problem to improve our bounds.

Acknowledgements

We gratefully acknowledge helpful conversations with John Gilbert, Tom Leighton, John Reif, and David Shmoys.

References

[1] George, J. A., "Nested Dissection of a regular finite element mesh", *SIAM Journal on Numerical Analysis 10* (1983), pp. 345-367.

[2] J. R. Gilbert, "Some Nested Dissection Order is Nearly Optimal", *Information Processing Letters* 26 (1987/88), pp. 325-328.

[3] J. R. Gilbert, personal communication (1989).

[4] J. R. Gilbert and H. Hafsteinsson, "Approximating treewidth, minimum front size, and minimum elimination tree height", manuscript, 1989

[5] J. R. Gilbert, D. J. Rose, and A. Edenbrandt, "A separator theorem for chordal graphs", *SIAM J. Alg. Disc. Meth. 5* (1984), pp. 306-313.

[6] U. Kjærulff, "Triangulation of graphs - Algorithms giving small total state space", *R 90-09, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg* (1990).

[7] P. Klein, C. Stein, and É. Tardos, "Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities", *Proceedings, 22nd ACM Symposium on Theory of Computing* (1990), pp. 310-321.

[8] F. T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms", *Proceedings, 29th Symposium on Foundations of Computer Science* (1988), pp. 422-431.

[9] F. T. Leighton, F. Makedon, and S. Tragoudas, personal communication, 1990

[10] R. J. Lipton, D. J. Rose, and R. E. Tarjan, "Generalized nested dissection", *SIAM Journal on Numerical Analysis 16* (1979), pp. 346-358.

[11] R. J. Lipton and R. E. Tarjan, "Applications of a planar separator theorem, *SIAM Journal on Computing 9* (1980), pp. 615-627.

[12] D. J. Rose, "Triangulated graphs and the elimination process", *Journal of Math. Anal. Appl. 32* (1970), p. 597-609.

[13] D. J. Rose, R. E. Tarjan, and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs", *SIAM J. Comp. 5* (1976), pp. 266-283.

[14] R. Schreiber, "A new implementation of sparse Gaussian elimination", *ACM Trans. on Mathematical Software 8:3* (1982), pp. 256-276.

[15] M. Yannakakis, "Computing the minimum fill-in is NP-complete", *SIAM J. Algebraic and Discrete Methods 2* (1981), pp. 77-79.