Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree

Hans L. Bodlaender* John R. Gilbert† Hjálmtýr Hafsteinsson‡

Ton Kloks§

Revised October 26, 1993

Abstract

Various parameters of graphs connected to sparse matrix factorization and other applications can be approximated using an algorithm of Leighton et al. that finds vertex separators of graphs. The approximate values of the parameters, which include minimum front size, treewidth, pathwidth, and minimum elimination tree height, are no more than $O(\log n)$ (minimum front size and treewidth) and $O(\log^2 n)$ (pathwidth and minimum elimination tree height) times the optimal values. In addition, we show that unless $P = NP$ there are no absolute approximation algorithms for any of the parameters.

Keywords: Graph separators, treewidth, pathwidth, multifrontal methods, sparse matrix computation, elimination trees, approximation algorithms, k-trees.

AMS(MOS) subject classifications: 05C50, 05C75, 65F50.

Computing Reviews descriptors: F.2.2, G.1.3, G.2.2.

---

*Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB The Netherlands. The research of this author was partially supported by the ESPRIT II Basic Research Actions of the EC under Contract No. 3075 (project ALCOM).

†Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304, USA.

‡Department of Computer Science, University of Iceland, 101 Reykjavík, Iceland.

§Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. This research was done while this author was working at the Department of Computer Science, Utrecht University. The research of this author was supported by the Foundation for Computer Science (SION) of the Netherlands Organization for Scientific Research (NWO).

1 Introduction

Many problems in science and engineering require the solution of systems of linear equations. As the problems get larger, it becomes important to exploit the fact that many such systems are sparse. Often each equation only involves a few of the variables. By taking advantage of sparsity it is often possible to solve substantially larger linear systems.

To solve the symmetric positive definite linear system $Ax = b$ via Cholesky factorization, one first computes the Cholesky factor L such that $A = LL^T$, and then solves the triangular systems $Ly = b$ and $L^T x = y$. If A is sparse, one may first permute the rows and columns of the matrix symmetrically, thus solving $(PAP^T)Px = Pb$ for some permutation matrix P. The permutation is typically chosen to try to make the factorization more efficient according to several measures of complexity. The permutation matrix corresponds to a reordering of the vertices of the graph $G(A)$ of the matrix.

Several different parameters of the graph $G(A)$ dictate how efficiently we can solve the linear system. Among these parameters are treewidth, minimum front size, minimum maximum clique, and minimum elimination tree height. Small *front size* is important in the multifrontal algorithm [8, 22]: front size is related to the amount of fast memory needed for factorization. *Elimination tree height* measures the parallel time needed to factor A by Gaussian elimination with unlimited processors. All these parameters depend on the ordering on the rows and columns of A. Unfortunately, determining the orderings that give the optimal values of these parameters is NP-complete [2, 9, 25]. Therefore we have to be content with approximations.

The main point of this paper is that minimum front size and elimination tree height are intimately related to three other graph parameters, namely *treewidth*, *pathwidth*, and the size of *separators* in subgraphs of the graph, and that all these parameters can be approximated within a polylogarithmic factor of optimal in polynomial time. Treewidth has several other applications to graph algorithms and to graph minor theory [1]. Pathwidth has important applications in the theory of VLSI layout, and is equivalent to several other graph parameters, including minimum chromatic number of a containing interval graph and node search number. The pathwidth problem is also equivalent to the gate matrix layout problem. See [24] for an overview.

In Section 3 we show that optimal treewidth, pathwidth, elimination

Table 1: Parameters of graph $G = (V, E)$ or symmetric sparse matrix A.

k-tree number	smallest k such that G is a subgraph of a k-tree
treewidth	minimum width of a tree decomposition of G
pathwidth	minimum width of a path decomposition of G
separator number	max (over subsets of V) of smallest 1/2-vertex separator
min etree height	min (over elimination orders for A) elimination tree height
min frontsize	min (over elimination orders for A) of largest frontal matrix
min max clique	min (over chordal completions of G) of largest clique in G^+

tree height, front size, minimax clique, and separator number are all within $O(\log n)$ of each other. In Section 5 we show that all these parameters can be approximated efficiently by using a recent result of Leighton et al. (see Lemma 4.1 in [16], and also [19]) on approximating graph separators. The approximations are no more than $O(\log n)$ or $O(\log^2 n)$ times the optimal values. (Some of these results have been obtained independently by Klein et al. [16].) Finally, we show in Section 6 that none of these parameters can be approximated within an additive constant of optimal in polynomial time unless $P = NP$.

2 Definitions

In this section we define the various graph parameters which we relate to each other in Section 3 and approximate in Section 5. Table 1 lists the parameters. We assume that the reader is familiar with standard graph theoretic notation (see Harary [14], for example). The subgraph of $G = (V, E)$ *induced* by $W \subseteq V$ is denoted by $G[W]$. Logarithms are taken to base 2 unless otherwise specified.

The first set of definitions concern parameters of graphs that have been studied in graph minor theory [29].

The class of *k-trees* is defined recursively as follows. The complete graph on k vertices is a k-tree. A k-tree with $n + 1$ vertices ($n \geq k$) can be constructed from a k-tree with n vertices by adding a vertex adjacent to all vertices of one of its k-vertex complete subgraphs, and only to these vertices. A *partial k-tree* is a graph that contains all the vertices and a subset of the edges of a k-tree. The smallest k such that G is a partial k-tree is its *k-tree* *number.*

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where T is a tree and $\{X_i\}$ is a collection of subsets of V, such that

- $\bigcup_{i \in I} X_i = V$.
- For all $(v, w) \in E$, there exists an $i \in I$ with $v, w \in X_i$.
- For all $i, j, k \in I$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The third condition can be replaced by the equivalent condition that $\{i \in I | v \in X_i\}$ forms a connected subtree of T for all $v \in V$.

The *treewidth* of a tree decomposition $(\{X_i\}, T)$ is $\max |X_i| - 1$. The treewidth of a graph is the minimum treewidth over all possible tree decompositions of that graph. The problem of finding the treewidth of a given graph G is NP-complete [2]. However, many NP-complete graph problems can be solved in polynomial and even linear time if restricted to graphs with constant treewidth (see e.g. [3, 4]). For any fixed k, determining whether the treewidth of G is at most k (and finding a corresponding tree decomposition) can be done in $O(n)$ time [6].

A *path decomposition* of a graph $G = (V, E)$ is a tree decomposition $(\{X_i\}, T)$ such that T is a path. The *pathwidth* of such a path decomposition is $\max |X_i| - 1$. The pathwidth of a graph is the minimum pathwidth over all possible path decompositions of that graph. The notion of pathwidth has several important applications, for example in VLSI layout theory [24].

The next few definitions measure the difficulty of splitting a graph approximately in half by deleting edges or vertices.

Let α be a constant between 0 and 1. An *α-vertex separator* of a graph $G = (V, E)$ is a set $S \subseteq V$ of vertices such that every connected component of the graph $G[V - S]$ obtained by removing S from G has at most $\alpha \cdot |V|$ vertices. An *α-edge separator* of G is a set $S \subseteq E$ of edges such that every connected component of the graph $(V, E - S)$ obtained by removing S from G has at most $\alpha \cdot |V|$ vertices.

For $W \subseteq V$, an *α-vertex separator of* W in $G = (V, E)$ is a set $S \subseteq V$ of vertices such that every connected component of the graph $G[V - S]$ contains at most $\alpha \cdot |W|$ vertices of W. An *α-edge separator of* W in G is defined similarly.

The following definition is not standard, but is useful for our purposes. The *separator number* of a graph $G = (V, E)$ is the maximum over all subsets W of V of the size of the smallest 1/2-vertex separator of W in G.

Thus the separator number measures the difficulty of separating the "hardest" subset of the vertices of G. Every subgraph of G has a 1/2-vertex separator of size at most the separator number of G.

Our final set of definitions concern a graph-theoretic model of symmetric Gaussian elimination that has been used extensively in sparse matrix computation [11].

The *elimination game* on a graph G repeats the following step until every vertex of G has been chosen (or "eliminated"). Choose a vertex v, and add edges to make adjacent all of the neighbors of v that have not yet been chosen. These added edges are called *fill edges*. The result of playing the elimination game is called the *filled graph* and written G^+; if π is the order in which the vertices were chosen, we sometimes write G^+_π. The elimination game models symmetric Gaussian elimination (or Cholesky factorization) on a matrix A whose graph is $G(A) = G$; the filled graph G^+ is the nonzero structure of the triangular factor. In this setting, choosing an order π for the vertices corresponds to choosing a permutation P for the rows and columns of the matrix A, and performing Gaussian elimination on PAP^T. The number of edges in G^+_π is called the *fill* due to ordering π, and measures the storage needed to hold the triangular factor; thus, choosing π to decrease fill is one goal in sparse factorization.

The filled graph G^+_π is a *perfect elimination graph*, which means that the elimination game can be played on G^+_π without causing any additional fill (using the order π, in fact). Perfect elimination graphs are the same as *chordal* graphs, which are graphs with no induced subgraphs that are cycles of at least four vertices [30]. A chordal graph obtained by adding edges to G is called a *chordal completion* of G; for all G and π, the graph G^+_π is a chordal completion of G, and every minimal chordal completion of G is G^+_π for some π [31].

Let $C_\pi(v)$ be the set of neighbors of vertex v that are still unchosen when v is eliminated; in other words, the set of neighbors of v in G^+_π that are higher-numbered than v according to π. The *elimination tree* T_π has the same vertices as G, and has a parent relation defined as follows [23, 32]: the parent of vertex i is the lowest-numbered vertex of $C_\pi(i)$, or equivalently the smallest of the higher-numbered neighbors of i in G^+_π. The elimination tree is a depth-first spanning forest of G^+_π, with one connected component for each connected component of G (or equivalently of G^+_π) [23]. The elimination tree of $G(A)$ describes the dependencies between the columns of the matrix A during Cholesky factorization: the children of a vertex must be computed

$$F = \begin{bmatrix} d & \mathbf{v}^T \\ \mathbf{v} & B \end{bmatrix} = \begin{bmatrix} \sqrt{d} & \mathbf{0} \\ \mathbf{v}/\sqrt{d} & I \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & B - \mathbf{v}\mathbf{v}^T/d \end{bmatrix} \begin{bmatrix} \sqrt{d} & \mathbf{v}^T/\sqrt{d} \\ \mathbf{0} & I \end{bmatrix}$$

Figure 1: A step in the multifrontal algorithm.

before their parent. The height of the elimination tree is thus a measure of the parallel time to factor A with unlimited processors. Generally, different permutations π give trees T_π of different heights. Thus one goal in parallel Cholesky factorization is to reorder the rows and columns of A to reduce the height of the elimination tree. The height of the shortest elimination tree (over all choices of π) is the *min etree height*.

The *front size* of G_π is the maximum size of $C_\pi(v)$ over all v. Like elimination tree height, front size depends on the choice of ordering π. The smallest front size of G over all π is the *min front size*. Reordering to decrease front size is important for the *multifrontal* algorithm [8, 22]. The multifrontal algorithm computes the Cholesky factorization of a sparse matrix by doing sequence of partial factorizations of small dense matrices, the goal being to make better use of hierarchical storage, vector floating-point hardware, or sometimes parallelism. Figure 1 shows one elimination step of the method: here $\mathbf{v}$ is only the nonzeros below the diagonal in the column being eliminated, and the *frontal matrix* F contains only the rows and columns corresponding to nonzeros in the column being eliminated. The *update matrix* $B - \mathbf{v}\mathbf{v}^T/d$ is dense, and is saved for use in later elimination steps. Many such matrices may be saved at the same time, but only enough main memory for one frontal matrix is needed. The front size of A is the dimension of the largest update matrix in the multifrontal factorization of A, or equivalently one less than the dimension of the largest frontal matrix. The front size of A can also be characterized as the largest number of nonzeros below the diagonal in any column of its Cholesky factor.

Finally, we define the *min max clique* of a graph G to be the the minimum, over all elimination orders π, of the size of the largest clique in the filled graph G_π^+. Equivalently, it is the smallest clique number of any chordal completion of G.

3 Relationships

In this section we prove several lemmas showing that the parameters listed in Table 1 are closely related. Theorem 12 at the end of the section summarizes these relationships. Some of the results are known, but we present them all here in order to show how closely linked these parameters are, and to demonstrate how a separator approximation algorithm can be used to approximate all the different parameters. The results relating elimination tree height to treewidth, pathwidth, and separator number are new. Several of the other relationships were already known, often only as 'folklore', and are often hard to trace to their origins.

Lemma 1 *For $k \geq 1$, the treewidth of G is at most k if and only if G is a partial k-tree; thus, treewidth is equal to k-tree number.*

Proof: See, for example, van Leeuwen [34]. $\Box$

Lemma 2 *The treewidth of G is equal to the minimum, over all elimination orders π, of the maximum, over all vertices v, of the number of higher-numbered neighbors of v in G_π^+.*

Proof: See, for example, Arnborg [1]. $\Box$

Lemma 3 *If A is a symmetric positive definite matrix and $G = G(A)$, then the treewidth of G is equal to the min front size of A.*

Proof: This is just a restatement of Lemma 2. $\Box$

Lemma 4 *Graph G has treewidth k iff the minimum over all elimination orders π of the size of the largest clique in G_π^+ is $k+1$. Thus, treewidth = min max clique $-$ 1.*

Proof: Let G have treewidth k. By Lemma 2, for every elimination order π there is a vertex v such that the set $C_\pi(v)$ of higher-numbered neighbors of v has size at least k. In the elimination game, the set $C_\pi(v) \cup \{v\}$ (which has size $k+1$) becomes a clique in G_π^+. Therefore min max clique $\geq$ treewidth$+1$.

Conversely, if min max clique is $k+1$, then for every elimination order π there is a clique of size at least $k+1$ in G_π^+. The lowest-numbered vertex in that clique has at least k higher-numbered neighbors in G_π^+, so the front size is at least k. By Lemma 3, then, min max clique $\leq$ treewidth $+ 1$. $\Box$

Now we show that separators are related to the other parameters. The following lemma is probably the oldest separator theorem; a version is due to C. Jordan in 1869 [15, 17], and Lewis et al. [20] rediscovered it in the early 1960s.

Lemma 5 *Let T be a tree with n vertices, and let W be a subset of the vertices. There is a vertex v in T such that every component of $T - v$ contains at most $\frac{1}{2}\,|W|$ vertices of W. Such a vertex can be found in $O(n)$ time.*

Proof: Let $\delta = \frac{1}{2} \mid W \mid$. The following algorithm finds v. Choose v arbitrarily to start. If no component of $T - v$ contains more than δ vertices of W, we are done. Otherwise, let v' be the neighbor of v in the component of $T - v$ that contains more than δ vertices of W. Replace v by v' and repeat. Notice that the component of $T - v'$ containing v has less than δ vertices of W, so the algorithm never repeats a choice of v and eventually terminates. □

In our terminology, this says that the separator number of a tree is 1. The next result generalizes from trees to partial k-trees. Several statements equivalent to Lemma 6 have appeared [13, 29]; this short proof is similar to one due to Liu [23].

Lemma 6 *Let $G = (V, E)$ be a graph with treewidth at most k. Let $W \subseteq V$. Then there exists a $\frac{1}{2}$-vertex separator of W in G of size at most $k + 1$.*

Proof: Consider an elimination order π such that G_π^+ has front size at most k. Use Lemma 5 to find vertex v which is a $\frac{1}{2}$-vertex separator of W in the elimination tree T_π. Because the elimination tree is a depth-first search tree of G_π^+ [23], every edge of G joins an ancestor to a descendant in T_π. Let S' be the set of proper ancestors of v in T_π that are adjacent (in G) to descendants of v in T_π. Then $S = S' \cup \{v\}$ is a $\frac{1}{2}$-vertex separator of W in G (and, in fact, in G_π^+). We see that S' is equal to $C_\pi(v)$, and hence S has size at most $k + 1$. □

The next pair of lemmas relates min etree height to vertex separator number. The construction in Lemma 7 is from Gilbert [12], where it was used to prove bounds on fill.

Lemma 7 *If G and its subgraphs have α-vertex separators of size at most s, then some ordering gives an elimination tree of height at most $s \log_{1/\alpha} n$.*

Proof: Let π be George's nested dissection ordering [10] for G: find an α-vertex separator S of size at most s, order the vertices of the separator last, and order the connected components of $G - S$ recursively by nested dissection. If the successive vertices of some path in G_π^+ have monotone increasing values in the ordering π, then the path can only include vertices from one separator on each level of recursion. There are at most $\log_{1/\alpha} n$ levels, so the longest such path is at most $s \log_{1/\alpha} n$. Any path from a leaf to the root of the elimination tree is monotone, so the lemma follows. $\Box$

Taking $\alpha = \frac{1}{2}$ in Lemma 7, we conclude that min etree height is at most $\log n$ times separator number. Now we show that separator number is at most min etree height.

Lemma 8 *If G has an elimination tree of height $h > 0$, then every subset W of vertices of G has a $\frac{1}{2}$-vertex separator of size at most h.*

Proof: Let S be the $\frac{1}{2}$-vertex separator for W in G that is found in Lemma 6. All the vertices of S lie on a path from a leaf to the root of the tree. Thus the size of S is at most $h + 1$. If S has $h + 1$ vertices, then it includes all the vertices on a path from some leaf v to the root. In this case either $S - v$ is a $\frac{1}{2}$-vertex separator for W, or $W = \{v\}$ and W itself is the desired separator. $\Box$

Next we relate the minimum elimination tree height to the minimum size of the maximum clique in G^+.

Lemma 9 *The separator number of G is less than or equal to min max clique.*

Proof: This follows directly from Lemma 4 and Lemma 6. $\Box$

Lemma 10 *If the minimum maximum clique of G^+ has size k, then the shortest elimination tree of G is lower than $k \log n$.*

Proof: This is immediate from Lemma 7 and Lemma 9. $\Box$

Now we relate pathwidth to the other parameters. As path decompositions are a special case of tree decompositions, the treewidth of a graph is never larger than the pathwidth. We also have the following interesting relationship.

Lemma 11 *If G has an elimination tree with height k, then the pathwidth of G is at most k.*

Proof: Let π be an ordering such that the elimination tree T_π has height k. Number the leaves of T_π as $v_1, \ldots, v_r$, from left to right. For $1 \leq j \leq r$, let X_j consist of v_j and all ancestors of v_j in T_π. Let P be the path $(\{1, 2, \ldots, r\}, \{(i, i+1) \mid 1 \leq i < r\})$. Then $(\{X_i\}, P)$ is a path decomposition of the filled graph G_π^+, and hence of G, with pathwidth k. □

As a direct consequence we have that the treewidth of G is no larger than the height of an elimination tree and the minimum maximum clique of G^+ is no larger than the height plus one.

Finally, we summarize all these relationships in a theorem.

Theorem 12 *Let G be a graph with at least one edge, and let A be a symmetric positive definite matrix with $G(A) = G$. The minimum front size of A is equal to the treewidth of G and to the smallest k such that G is a partial k-tree, and this number is one less than the smallest clique number of any chordal completion of G. The minimum elimination tree height of A is no less than these three parameters, and is at most $\log n$ times them. The pathwidth of G lies between treewidth and minimum elimination tree height. Minimum etree height is also between the separator number of G and $\log n$ times the separator number, but the separator number can be up to a factor of $\log n$ less or more than treewidth, minimum front size and minimum maximum clique. In other words,*

- *treewidth = min partial k-tree = min frontsize = min max clique − 1.*
- *separator number $-1 \leq$ treewidth $\leq$ pathwidth $\leq$ min etree height $\leq$ separator number $\cdot \log n$.*

□

4 Approximation of vertex separators

Leighton and Rao [19] have obtained approximation algorithms for various separator problems, including the problem of finding minimum size balanced edge separators. Using these results, Leighton and Rao obtained similar results for vertex separators, including the following result, upon which our algorithms depend heavily.

Theorem 13 ([19]) *There exists a polynomial algorithm that, given a graph $G = (V, E)$ and a set $W \subseteq V$, finds a $\frac{2}{3}$-vertex separator $S \subseteq V$*

of W in G of size $O(w \cdot \log n)$, where w is the minimum size of a $\frac{1}{2}$-vertex separator of W in G.

When we now apply Lemma 6, we get the following result, which is the fundamental step in our approximation algorithm.

Theorem 14 *There exist a constant $\beta \geq 1$ and a polynomial time algorithm that, given a graph $G = (V, E)$ and a set $W \subseteq V$, finds a $\frac{2}{3}$-vertex separator of W in G of size $\beta \cdot \log n \cdot k$, where $n = |V|$ and k is the treewidth of G.*

Proof: Lemma 6 implies that there exists a $\frac{1}{2}$-vertex separator of W in G of size $k+1$. The result follows by using the algorithm of Theorem 13, taking β to be enough larger than the constant hidden in the O of Theorem 13 to account for the factor $(k+1)/k$. □

In the remainder of the paper, we take β to be the constant from this theorem.

5 Approximation algorithms

In this section we give a polynomial time approximation algorithm for the treewidth problem that is at most a factor of $O(\log n)$ off optimal. From the analysis in Section 3, this directly implies polynomial time approximations for minimum maximum cliques and minimum front size that are a factor of $O(\log n)$ off optimal, and for minimum height elimination trees that are a factor of $O(\log^2 n)$ off optimal. Readers familiar with the approximation algorithms for constant treewidth of Lagergren [18], of Reed [26], or of Robertson and Seymour [28] may note some similarities. Our algorithm also has some resemblance to Lipton, Rose, and Tarjan's version of nested dissection [21].

Our approximation algorithm consists of calling *makedec*$(V, \emptyset)$, where *makedec* is the recursive procedure shown in Figure 2.

Lemma 15 *If Z and W are disjoint sets of vertices of G, then* makedec(Z, W) *returns a tree decomposition of $G[Z \cup W]$ such that the root node of the tree decomposition contains all vertices in W. If $|W| \leq 6\beta k \log n$, where k is the treewidth of G and $n = |V|$, then the treewidth of this tree decomposition is at most $8\beta k \log n$.*

Proof: We prove this by induction on the recursive structure of the *makedec* procedure. Clearly the claim is true in case $3 \cdot |Z| \leq |W|$.

procedure *makedec*(Z, W);

(*Comment:* Z *and* W *are disjoint sets of vertices*.)

if $3 \cdot |Z| \leq |W|$ **then**

Return a tree decomposition with one single node, containing $Z \cup W$;

else

Find a 2/3-vertex separator S of W in $G[Z \cup W]$,

using the algorithm of Theorem 14;

Find a 2/3-vertex separator S' of $Z \cup W$ in $G[Z \cup W]$,

using the algorithm of Theorem 14;

Let $G_1, \ldots, G_t$ be the connected components of $G[Z \cup W - (S \cup S')]$;

for $i \leftarrow 1$ **to** t **do**

Let $Z_i \leftarrow$ the vertices of G_i in Z;

Let $W_i \leftarrow$ the vertices of G_i in W;

call *makedec*$(Z_i, W_i \cup S \cup S')$;

end for

Return the following tree decomposition:

Take a new root node $r_{Z,W}$, containing vertices $W \cup S \cup S'$;

Then add all tree decompositions returned by the recursive calls,

with an edge from the root of each to $r_{Z,W}$;

end if

end procedure

Figure 2: Algorithm to compute tree decomposition.

We first show that for all edges $(v,w) \in E$ with $v,w \in Z \cup W$, there exists a node i in the tree decomposition with $v,w \in X_i$. Suppose this is not the case. First consider an edge $(v,w) \in E$ with $v,w \in Z \cup W$. If v and w both are in the set W, then $v,w \in X_{r_{Z,W}}$. Otherwise, v and w both belong to a set $Z_i \cup W_i \cup S \cup S'$. By induction, there is a set X_j in the tree decomposition returned by *makedec*$(Z_i, W_i \cup S \cup S')$ with $v,w \in X_j$.

We now show that $\{i \in I \mid v \in X_i\}$ forms a connected subtree in the decomposition-tree for all $v \in Z \cup W$. If $v \notin X_{r_{Z,W}}$, then this holds by induction, as v then belongs to exactly one set Z_i. Otherwise, for each of the subtrees under $r_{Z,W}$, either v does not appear in any of the nodes in this subtree, or the nodes containing v form, by induction, a connected subtree of this subtree, and include the root of this subtree, i.e., the child of $r_{Z,W}$ that is in this subtree. The result now follows. Therefore the procedure indeed outputs a tree decomposition of $G[Z \cup W]$.

We now have to show that the treewidth of the tree decomposition is at most $8\beta k \log n$. By induction, it is sufficient to show that $|X_{r_{Z,W}}| \leq 8\beta k \log n$, and that $|W_i \cup S \cup S'| \leq 6\beta k \log n$. The first inequality follows by using the assumption on the size of W, and then using Theorem 14 to bound the size of S and S'. The second inequality follows because $S \cup S'$ is an $\frac{2}{3}$ separator of W in $G[Z \cup V]$, and hence each W_i is of size at most $2/3|W| \leq 4\beta k \log n$, whence $|W_i \cup S \cup S'| \leq 6\beta k \log n$. □

Thus, we have obtained the following result:

Theorem 16 *There exists a polynomial time algorithm that, given a graph $G = (V,E)$ with $|V| = n$, finds a tree decomposition of G with treewidth at most $O(k \log n)$, where k is the treewidth of G.* □

This result implies approximation algorithms for the other parameters discussed in this paper. Clearly, by Lemmas 3 and 4 we have also a polynomial time algorithm that, given a graph G, solves the minimum maximum clique problem and the minimum front size problem within $O(\log n)$ times optimal. We also have:

Theorem 17 *There exists a polynomial time algorithm that, given a graph $G = (V,E)$ with $|V| = n$, finds an elimination tree of G with height at most $O(h \log^2 n)$, where h is the minimum height of an elimination tree of G.*

Proof: Find a tree decomposition of G with the algorithm of Theorem 16. Then construct an elimination order as in Lemma 7. We obtain an elimination tree of G with height at most $\log n \cdot O(\log n) \cdot k$, where k is the treewidth of G. Observe that k is smaller than or equal to the pathwidth of G; hence, by Lemma 11, k is at most h. $\Box$

Similarly we can obtain:

Theorem 18 *There exists a polynomial time algorithm that, given a graph $G = (V, E)$ with $|V| = n$, finds a path decomposition of G with pathwidth at most $O(k \log^2 n)$, where k is the pathwidth of G.* $\Box$

6 Absolute approximations

In this section we show that if $P \neq NP$, then no absolute approximation algorithms exist for the minimum height elimination tree problem, for treewidth (and hence for minimum front size and min max clique), or for pathwidth.

Given an approximation algorithm $\mathcal{A}$ for a minimization problem, we can distinguish between three kinds of performance guarantees. First, in an *absolute approximation*, the approximate solution $\mathcal{A}(I)$ is within an additive constant of the optimal solution $\mathcal{OPT}(I)$. Second, the approximate solution can be within a multiplicative constant of the optimal one. Finally, the ratio between the optimal and approximate solutions can grow with the size of the problem. The algorithms we have presented above all have performance guarantees of the third kind, with ratios of $O(\log n)$ or $O(\log^2 n)$. The hardest of these bounds to achieve is the absolute bound; few NP-complete problems have absolute approximation algorithms. We first prove that the minimum height elimination tree problem has no absolute approximation algorithm unless $P = NP$.

Theorem 19 *If $P \neq NP$ then no polynomial time approximation algorithm $\mathcal{A}$ for the minimum height elimination tree problem can guarantee $\mathcal{A}(I) - \mathcal{OPT}(I) \leq K$ for any constant K.*

Proof: Assume we have a polynomial time absolute approximation algorithm $\mathcal{A}$, so that $\mathcal{A}$ always gives an elimination tree with height at most K more than optimal. We will show that then we can solve the *mutual independent set* problem (MUS) in polynomial time. The MUS problem is the following: Given a bipartite graph $B = (P, Q, E)$, are there sets $V_1 \subseteq P$ and $V_2 \subseteq Q$ with $|V_1| = |V_2| = k$, such that no edge joins a vertex in V_1 to a vertex in V_2? Pothen [25] shows that this problem is NP-complete.

Let $B = (P, Q, E)$ be a bipartite graph, with vertex sets P and Q and edge set E. Graph B is a *chain graph* if the adjacency sets of vertices in P form a chain, that is, if the vertices of P can be ordered so that

$$\mathrm{Adj}(v_1) \supseteq \mathrm{Adj}(v_2) \supseteq \cdots \supseteq \mathrm{Adj}(v_p).$$

The *biclique* $C = (P, Q, E \cup P^2 \cup Q^2)$ of B is the graph that results from adding edges to B to make each of P and Q into cliques.

Yannakakis [35] has shown that if we add edges to the bipartite graph B to make it a chain graph B', then adding the same edges to B's corresponding biclique C makes it a chordal graph C'. The graph C' is called a *chordal completion of the biclique* C. Pothen [25] has proved that B has mutually independent sets of size k iff there exists a chordal completion C' of B with elimination tree of height $n - k - 1$, or, in other words, iff the min etree height of B' is $n - k - 1$.

Let $B = (P, Q, E)$ be given. Construct a new bipartite graph $\hat{B} = (P_1 \cup \cdots \cup P_{K+1}, Q_1 \cup \cdots \cup Q_{K+1}, \hat{E})$ that contains $K + 1$ copies of B and additional edges between the copies. If there is an edge between vertices v and w in B ($v \in P$, $w \in Q$), then $\hat{B}$ has an edge between v_i and w_j ($v_i \in P_i$ and $w_j \in Q_j$), for $i, j = 1, \ldots, K + 1$. The new graph $\hat{B}$ has $(K + 1)n$ vertices and $(K + 1)^2 m$ edges. In Figure 3 we show $\hat{B}$ when $K = 1$.

Let $\hat{B}'$ be the biclique of $\hat{B}$. Now $\hat{B}$ has mutually independent sets of size $(K+1)k$ iff the minimum etree height of $\hat{B}'$ is at most $(K+1)(n-k)-1$.

Now, if B has a mutual independent set of size k, then $\hat{B}$ has a mutual independent set of size $(K+1)k$, so the minimum etree height of $\hat{B}'$ is at most $(K+1)n-(K+1)k-1$. If every mutual independent set of B has size $\leq k-1$, then every mutual independent set of $\hat{B}$ has size at most $(K+1)(k-1)$ (we can choose at most $2 \times (k-1)$ vertices in each pair (P_i, Q_i)), hence the minimum etree height of $\hat{B}'$ is at least $(K+1)n - (K+1)(k-1) - 1$. So, B has a mutual independent set of size k, if and only of algorithm $\mathcal{A}$ outputs that $\hat{B}'$ has minimum etree height at most $(K+1)n - (K+1)k - 1 + K$. So, we have a polynomial time algorithm for the NP-complete MUS problem, contradiction our assumption that $P \neq NP$. □

A similar result can be proven for the treewidth problem. We need the following lemma.

Lemma 20 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of $G = (V, E)$. Let $W \subseteq V$ be a clique in G. Then there exists an $i \in I$ with $W \subseteq X_i$.*

Proof: See, for example, Bodlaender and Möhring [5]. □

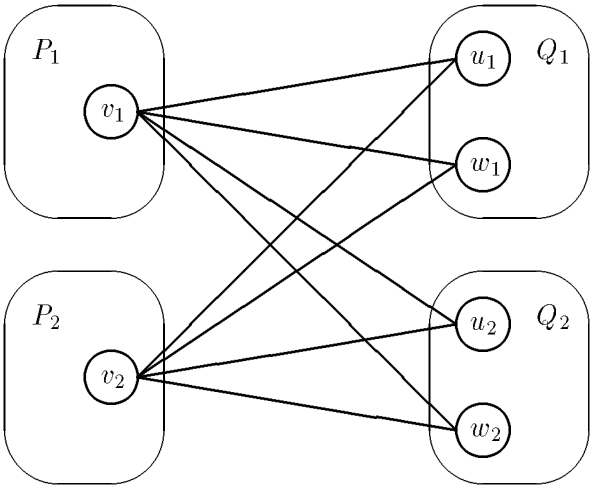

Figure 3: The graph $\hat{B}$ when $K = 1$

Theorem 21 *If $P \neq NP$ then no polynomial time approximation algorithm $\mathcal{A}$ for the treewidth problem (and hence for minimum front size, and minimum maximum clique) can guarantee $\mathcal{A}(G) - \mathcal{OPT}(G) \leq K$ for a fixed constant K.*

Proof: Assume we have a polynomial time algorithm $\mathcal{A}$ that, given a graph $G = (V, E)$, finds a tree decomposition of G with treewidth at most K larger than the treewidth of G. Let a graph $G = (V, E)$ be given. Let $G' = (V', E')$ be the graph obtained by replacing every vertex of G by a clique of $K+1$ vertices, and adding edges between every pair of adjacent vertices in G. Thus $V' = \{v_i \mid v \in V, 1 \leq i \leq K+1\}$ and $E' = \{(v_i, w_j) \mid (v = w \wedge i \neq j) \vee (v, w) \in E\}$. We examine the relationship between the treewidth of G and the treewidth of G'.

Suppose we have a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G with treewidth L. One easily checks that $(\{Y_i \mid i \in I\}, T = (I, F))$ with $Y_i = \{v_j \mid v \in X_i, 1 \leq j \leq K+1\}$ is a tree decomposition of G' with treewidth $(L+1)(K+1)-1$. It follows that treewidth$(G') \leq$ (treewidth$(G)+1) \cdot (K+1) - 1$.

Next suppose we have a tree decomposition $(\{Y_i \mid i \in I\}, T = (I, F))$ of G' with treewidth M. Let $X_i = \{v \in V \mid \{v_1, v_2, \cdots, v_{K+1}\} \subseteq Y_i\}$. We claim that $(\{X_i \mid i \in I\}, T = (I, F))$ is a tree decomposition of G with treewidth $(M+1)/(K+1) - 1$. Let $(v, w) \in E$. Note that $v_1, v_2, \cdots, v_{K+1}, w_1, w_2, \cdots, w_{K+1}$ form a clique in G'. Hence, by Lemma 20 there exists an $i \in I$ with $\{v_1, \cdots, v_{K+1}, w_1, \cdots w_{K+1}\} \subseteq Y_i$, and thus $v, w \in X_i$. Let $j \in I$ be on the path in T from $i \in I$ to $k \in I$. If $v \in X_i \cap X_k$, then $\{v_1, \cdots, v_{K+1}\} \subseteq Y_i \cap Y_k$, and hence by definition of tree decomposition $\{v_1, \cdots, v_{K+1}\} \subseteq Y_j$, so $v \in X_j$. Clearly, $\max_{i \in I} |X_i| \cdot (K+1) \leq \max_{i \in I} |Y_i|$. This finishes the proof of our claim. It follows that treewidth$(G') \geq$ (treewidth$(G) + 1) \cdot (K+1) - 1$, and hence that treewidth$(G') =$ (treewidth$(G) + 1) \cdot (K+1) - 1$.

Now we can describe the polynomial time algorithm for the treewidth problem. Let G be the input graph. Form G', and apply algorithm $\mathcal{A}$ to G'. Apply the construction described above to form a tree decomposition of G. This must be a tree decomposition with minimum treewidth: if the treewidth of G is k, then the treewidth of G' is $(k+1)(K+1)-1$. Hence, $\mathcal{A}$ outputs a tree decomposition of G' with treewidth at most $(k+1)(K+1) - 1 + K$, and the algorithm described above outputs a tree decomposition of G with treewidth at most $\lfloor ((k+1)(K+1) + K)/(K+1) - 1 \rfloor = k$. Thus we would have a polynomial time algorithm for treewidth. □

In the same way we can prove the following theorem. This result was also proved (with different terminology) by Deo et al. [7].

Theorem 22 *If $P \neq NP$ then no polynomial time approximation algorithm $\mathcal{A}$ for the pathwidth problem can guarantee $\mathcal{A}(G) - \mathcal{OPT}(G) \leq K$ for a fixed constant K.* $\square$

It is possible to strengthen these results slightly.

Theorem 23 *If $P \neq NP$ then no polynomial time approximation algorithm $\mathcal{A}$ for the minimum height elimination tree, treewidth, or pathwidth problem can guarantee $\mathcal{A}(G) - \mathcal{OPT}(G) \leq n^\epsilon$, where ϵ is a constant with $0 < \epsilon < 1$, and n denotes the number of vertices of G.*

Proof: Suppose $\epsilon < (c-1)/c$. Basically, we use the same proofs as for the case that the additive term was a constant. Instead of taking $K+1$ copies of G, or replacing vertices by cliques of $K+1$ vertices, we now take n^c copies of G (min etree height) or replace vertices by cliques of size n^c (treewidth, pathwidth). $\square$

7 Conclusions

We have presented algorithms to find bounded approximations to various parameters of graphs and sparse matrices. More specifically, for treewidth, minimum front size, and minimum maximum clique, we get approximations that are never more than $O(\log n)$ times optimal; for pathwidth and minimum height elimination tree we get approximations that are never more than $O(\log^2 n)$ times optimal. The key insight is that all these measures are tightly related to the size of separators in the graph.

An open problem is to find polynomial algorithms that give solutions that are only a constant times optimal for any of the parameters discussed in this paper. We have shown that none of the parameters can be approximated within an additive constant or term of the form n^ϵ for $\epsilon < 1$ of optimal unless $P = NP$.

Two other parameters of interest in sparse matrix computation are the minimum fill (or number of edges in chordal completion) over all elimination orders, and the minimum arithmetic operation count. Klein et al. [16] use a nested dissection algorithm somewhat similar to ours to give approximation algorithms for these measures that get within $O(\log^4 n)$ and $O(\log^6 n)$

times optimal (respectively) provided that the degree of the input graph is bounded by a constant.

Recently, Seymour and Thomas [33] have obtained a polynomial algorithm for the related notion of *branchwidth*, when restricted to planar graphs. As the branchwidth and treewidth of a graph differ by at most a factor of 1.5 [27], this gives a polynomial time approximation algorithm for treewidth of planar graphs with performance ratio 1.5, and polynomial time approximation algorithms for pathwidth and shortest elimination tree of planar graphs with performance ratio $O(\log n)$. An interesting question is whether a polynomial time algorithm exists that solves treewidth exactly on planar graphs.

Acknowledgement

Theorem 23 was suggested by a referee.

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey. *BIT*, 25:2–23, 1985.

[2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

[3] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[4] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11:631–644, 1990.

[5] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Meth.*, 6:181–188, 1993.

[6] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 226–234. ACM Press, 1993.

[7] N. Deo, M. S. Krishnamoorty, and M. A. Langston. Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer-Aided Design*, 6:79–84, 1987.

[8] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9:302–325, 1983.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

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

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

[12] J. R. Gilbert. Some nested dissection order is nearly optimal. *Information Processing Letters*, 26:325–328, 1988.

[13] J. R. Gilbert, D. J. Rose, and A. Edenbrandt. A separator theorem for chordal graphs. *SIAM Journal on Algebraic and Discrete Methods*, 5:306–313, 1984.

[14] F. Harary. *Graph Theory*. Addison-Wesley Publishing Company, 1969.

[15] C. Jordan. Sur les assemblages de lignes. *Journal Reine Angew. Math.*, 70:185–190, 1869.

[16] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 726–737. IEEE, 1990.

[17] D. König. *Theorie der Graphen*. Reprinted by Chelsea Publishing Company, New York, 1950.

[18] J. Lagergren. Efficient parallel algorithms for tree-decomposition and related problems. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 173–182. IEEE, 1990.

[19] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. Manuscript, submitted. Extended abstract appeared in: *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422–431. IEEE, 1988.

[20] P. M. Lewis II, R. E. Stearns, and J. Hartmanis. Memory bounds for the recognition of context-free and context-sensitive languages. In *IEEE Conference Record on Switching Theory and Logical Design*, pages 191–202, 1965.

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

[22] J. W. H. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *Siam Review*, 34:82–109, 1992.

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

[24] R. H. Möhring. Graph problems related to gate matrix layout and PLA folding. In G. Tinhofer et al., editor, *Computing Supplementum*, volume 7, pages 17–51. Springer-Verlag, 1990.

[25] A. Pothen. The complexity of optimal elimination trees. Technical Report CS–88–16, The Pennsylvania State University, 1988.

[26] B. Reed. Finding approximate separators and computing tree-width quickly. In *Proceedings of the 24th Annual Symposium on Theory of Computing*, pages 221–228, 1992.

[27] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decompositions. *J. Comb. Theory Series B*, 52:153–190, 1991.

[28] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. Manuscript, 1986.

[29] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[30] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32:597–609, 1970.

[31] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5:266–283, 1976.

[32] R. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Transactions on Mathematical Software*, 8:256–276, 1982.

[33] P. D. Seymour and R. Thomas. Call routing, rat catching, and planar branch width. In *DIMACS Workshop– Planar Graphs: Structures and Algorithms*. Center for Discrete Mathematics & Theorical Computer Science, Nov 1991.

[34] J. van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science. A: Algorithms and Complexity Theory*. North Holland, 1990.

[35] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2:77–79, 1981.