Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height

Hans L. Bodlaender John R. Gilbert
Hjálmtýr Hafsteinsson Ton Kloks

RUU-CS-91-1
January 1991

Utrecht University
Department of Computer Science
Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height

Hans L. Bodlaender John R. Gilbert
Hjálmtýr Hafsteinsson Ton Kloks

Technical Report RUU-CS-91-1
January 1991

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN:0924-3275

Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height

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

Abstract

We show how the value of 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 examine the existence of bounded approximation algorithms for the parameters, and show that unless $P = NP$, there are no absolute approximation algorithms for them.

1 Introduction

Many problems in science and engineering require the solving of linear systems of equations. As the problems get larger it becomes increasingly important to exploit the sparsity inherent in many such linear systems. Often each equation only involves a few of the variables. By taking advantage of that fact we are able to solve substantially larger linear systems. Solving the symmetric positive definite linear system $Ax = b$ via Cholesky factorization involves computing the Cholesky factor L, such that $A = LL^T$, and then solving the triangular systems $Ly = b$

*Department of Computer Science, University of Utrecht, P.O. Box 80.089, 3508 TB The Netherlands. The research of this author is 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, and University of Bergen, Norway.

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

§Department of Computer Science, University of Utrecht. The research of this author is supported by the Foundation for Computer Science (S.I.O.N.) of the Netherlands Organization for Scientific Research (N.W.O.) and by the ESPRIT II Basic Research Actions of the EC under Contract No. 3075 (project ALCOM).

and $L^T x = y$. If A is sparse we usually do some preprocessing on its associated graph, $G(A)$. Various parameters of this graph dictate how fast and efficiently we can solve the system. Among these parameters are treewidth, minimum front size, minimum maximum clique, and minimum elimination tree height. Having small front size is important in the multifrontal method [DR83, Liu90] and an ordering minimizing the elimination tree height minimizes the parallel time required to factor A. All the above 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 [ACP87, GJ79, Pot88]. Therefore we have to be content with approximations.

The notion of treewidth has several other applications (see e.g. [Arn85]). It is closely related to the pathwidth, which has among others important applications in the theory of VLSI layout. The pathwidth is equivalent to several other parameters of graphs, including the minimum chromatic number of an interval graph containing the graph as a subgraph and the node search number of a graph. The pathwidth problem is also equivalent to the gate matrix layout problem. See [Möh89] for an overview.

In this paper we will show how to use a recent result of Leighton et al. (see lemma 4.1 in [KARR90], and also [LR88]) on approximating graph separators to find approximations to the above parameters. These approximations will be no more than $O(\log n)$ or $O(\log^2 n)$ times the optimal values. Some of these results were obtained independently by Klein et al. [KARR90].

We will start with a few definitions. After that we explore the relationship between treewidth, pathwidth, minimum front size, minimum elimination tree height, and other related concepts. Then we present the result of Leighton et al. and our approximation algorithms. Finally we discuss bounded approximations for minimum elimination tree height, treewidth, and pathwidth.

2 Definitions

We assume that the reader is familiar with standard graph theoretic notation (see [Har69]). The subgraph of $G = (V, E)$ *induced* by $W \subseteq V$ is denoted by $G[W]$.

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.

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a collection of subsets of V, and T a tree, 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: for all $v \in V$, $\{i \in I | v \in X_i\}$ forms a connected subtree of T. The *treewidth* of a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph is the minimum treewidth over all possible tree-decompositions of that graph. It can be shown that G has treewidth at most k, if and only if G is a partial k-tree (see e.g. [vL90]).

The problem of finding the treewidth of a given graph G is NP-complete [ACP87]. 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. [ALS88, Bod90a].) For constant k, determining whether the treewidth of G is at most k, and finding a corresponding tree-decomposition can be done in polynomial time (see e.g. [Bod90b]). The first step of such an algorithm is to find a tree-decomposition of G which has not optimal, but still constant bounded treewidth [RS86b, Lag90].

A *path-decomposition* of a graph $G = (V, E)$ is a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$, such that T is a path. The *pathwidth* of a path-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. 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, e.g., in VLSI-layout theory (see [Möh89]).

The *elimination game* on a graph G repeats the following step until there are no more vertices. Pick a vertex v, delete it from the graph, and add edges between the neighbours of v that are not already adjacent. These added edges are called *fill edges*. The *filled graph* G^*_π is obtained by adding to G all the fill edges that occur when the elimination game is played using the order π on the vertices of G.

Let $C_\pi(v)$ be the set of uneliminated neighbours of vertex v when playing the elimination game with order π on the graph G. The treewidth of G can alternately be defined as the minimum over all orderings π of $\max_{v \in V} \mid C_\pi(v) \mid$. (See e.g. [Arn85].)

The *elimination tree* T is defined as follows. Vertex j is the parent of vertex i in T (with $j > i$) iff j is the lowest numbered among the higher numbered neighbours of i in the filled graph G^*.

The elimination tree T of a graph $G(A)$ describes the dependencies between the columns of the matrix A during column-oriented Cholesky factorization. If vertex i is the child of vertex j in T, then column i has to be computed before column j in A's Cholesky factorization. Consequently, we can compute all the columns on the same level in the tree simultaneously. Thus, the height of the elimination tree is a reasonable measure of the parallel time required to factor A. A completely dense matrix has a tree that is just one long chain, since each column depends on all the previous ones. In the case of sparse matrices the shape of the elimination

tree can vary. By reordering the rows and columns of A (equivalent to renumbering the vertices of $G(A)$) we can, to some degree, restructure the elimination tree. Thus the first step in parallel solution of sparse linear systems is to reorder the rows and columns of A in order to reduce the height of the elimination tree.

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

3 Relationships

Below we will show the relationships among the various parameters using a series of lemmas. Many of these results are not new, but we present them all here in order to demonstrate how closely linked these parameters are. In addition this will make it easier to see how the result of Leighton et al. can be used to find approximations to the different parameters.

Lemma 3.1 *A graph G has treewidth k iff the minimum, over all filled graphs G^* of G, of the largest clique in G^* is $k + 1$.*

Proof: In playing the elimination game the set $C_\pi(v) \cup \{v\}$ of v and its uneliminated neighbours becomes a clique in G^*. Thus if G has treewidth k (i.e. $\min_\pi \max_{v \in V} |C_\pi(v)| = k$) then the minimum over all filled graphs G^* of G, of the largest clique in G^* is at least $k + 1$. If we have a minimum maximum clique of size c then when its first vertex v is eliminated in the elimination game the rest of the vertices will become $C_\pi(v)$, and its size is $c - 1$. Thus the size of the minimum maximum clique is no more than $k + 1$ if the treewidth is k. □

The multifrontal method [DR83, Liu90] organizes the factorization of a sparse matrix into a 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 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, or one less than the dimension of the largest frontal matrix.

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

Figure 1: A step in the multifrontal method

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, or the maximum number of neighbors of any vertex when it is eliminated in the elimination game.

Lemma 3.2 *The minimum front size of a graph $G(A)$ with treewidth k over all orderings is k.*

Proof: If $G(A)$ has treewidth k, then there is some ordering π, such that the largest set $C_\pi(v)$ of higher numbered neighbours of any vertex v has size k. If we order the columns and rows of A according to π then at each step the nonzero elements of the vector $\mathbf{v}$ in Figure 1 correspond to the uneliminated neighbours of v in the elimination game. The frontal matrix has as many columns and rows as there are non-zeros in $\mathbf{v}$. Thus the largest frontal matrix has the same size as the largest $C_\pi(v)$, and the front size of $G(A)$ is the same as the treewidth. □

The following lemma, which can be traced back to C. Jordan (see [Kön36]) will be useful when proving things about the elimination tree.

Lemma 3.3 *Given a tree T we can, in time $O(n)$, find a vertex v such that $T \setminus \{v\}$ has no component of size greater than $\frac{n}{2}$.*

In other words, the algorithm finds an $\frac{1}{2}$ vertex separator of T of size 1. This result can easily be generalized as follows:

Lemma 3.4 *Given a tree T and a subset of the vertices W, there is a vertex v in T such that every component of $T \setminus \{v\}$ contains at most $\frac{1}{2}|W|$ vertices of W.*

Proof: Let $\delta = \frac{1}{2}|W|$. Consider the following algorithm. Start at any vertex c. If every component of $T \setminus \{c\}$ contains at most δ vertices of W we are done. Otherwise, let c' be the neighbour of c, in the component of $T \setminus \{c\}$ that contains more than δ vertices of W. Replace c by c' and repeat the process. Notice that the component of $T \setminus \{c'\}$ containing c has less than δ vertices of W. So the algorithm terminates. □

The following theorem, (see also e.g. [RS86a]), given here with a short proof, plays an important role in our approximation algorithm.

Theorem 3.5 *Let $G = (V, E)$ be a graph with treewidth $\leq k$. 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 tree T_π such that G_π^* has treewidth k. Pick a vertex v as indicated in the previous lemma. Let S be the set of ancestors of v adjacent to a vertex in subtree of T_π rooted at v (i.e. $S = C_\pi(v)$). This set is a separator of size less than or equal to k. This follows from the fact that $S \cup \{v\}$ forms a clique in G_π^*, and since the largest clique in G_π^* has size $k+1$, S cannot contain more than k vertices. So we can take $S \cup \{v\}$ as the required separator. □

The next pair of lemmas illustrates the relationship between elimination trees and vertex separators of graphs.

Lemma 3.6 *If G and its subgraphs have α vertex separators of size s, then there is an elimination tree of height $O(s \log n)$.*

Proof: If we apply the nested dissection ordering (see [Geo73]), i.e., order the vertices of the first separator last, then the vertices of the next level of separators, and so on, then the height of the resulting elimination tree is at most $s \log_{1/\alpha} n$. Since α is a constant, the height is $O(s \log n)$. □

Lemma 3.7 *If G has an elimination tree of height h then it and its subgraphs have $\frac{1}{2}$ vertex separators of size h.*

Proof: If we have an elimination tree T of height h, then we can use Lemma 3.3 to find a vertex v such that no component of $T \setminus \{v\}$ contains more than $\frac{n}{2}$ vertices. The set of ancestors of v adjacent to a vertex of the subtree rooted at v is a separator. Its size obviously cannot be more than the height of the elimination tree. This argument can be applied recursively to the remaining components, since the heights of their elimination trees cannot be more than h. □

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

Lemma 3.8 *If the minimum maximum clique of G^* has size k then the minimum height elimination tree of G is lower than $k \log n$.*

Proof: Assume that the minimum maximum clique of G^* has size k and it occurs when we use ordering π on the vertices. Consider the elimination tree T_π, which is the elimination tree of G using ordering π. We can pick a vertex v in T_π, such that no component of $T \setminus \{v\}$ contains more than $\frac{n}{2}$ vertices, using Lemma 3.3. let S be the set of ancestors of v adjacent to a vertex in subtree of T_π rooted at v. Then $S \cup \{v\}$ is a separator of size less than or equal to k, since this set forms a clique in G^* and the largest clique in G^* has size k. We continue recursively finding separators of size at most k in the components, none of which is larger than $\frac{n}{2}$. We can then use nested dissection to order the vertices, ordering the vertices of the first separator last, and so on. This will give us an elimination tree of height at most $k \log n$. (The same construction was used in [Gil88].) □

Now we give a relationship between pathwidth and 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 3.9 *If G has an elimination tree with height k, then the pathwidth of G is at most k.*

Proof: Number the leaves of the elimination tree $v_1, \ldots, v_r$, from left to right. Let X_j, $(1 \leq j \leq r)$ consist of v_j and all ancestors of v_j in the elimination tree. Now $(\{X_i \mid 1 \leq i \leq r\}, T = (\{1, 2, \cdots, r\}, \{(i, i+1) \mid 1 \leq i < r\})$ is a path-decomposition of the filled graph G^* 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.

Let us now summarize these relationships. Minimum front size is equal to the treewidth and minimum largest clique of G^* is one more than the treewidth of G. The minimum elimination tree height is no less than those three parameters, and at most $\log n$ times them. The pathwidth of a graph is "between" the treewidth of a graph, and its minimum elimination tree height. We can summarize these relationships as follows:

- treewidth = min front size = min max clique -1
- treewidth $\leq$ pathwidth $\leq$ min height $\leq$ treewidth $\cdot \log n$.

4 Approximations of vertex separators

In [LR88] Leighton and Rao present approximation algorithms for various separator problems, including the problem of finding minimum size balanced edge separators. Recently, Leighton et al. [Le90] obtained similar results for vertex separators, as reported in [KARR90]. We will use the following result.

Theorem 4.1 (Le90) *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 theorem 3.5, we get the following result, which is the fundamental step in our approximation algorithm.

Theorem 4.2 *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: Theorem 3.5 tells us that there exists a $\frac{1}{2}$ vertex separator of W in G of size $k+1$. The result hence follows by using the algorithm of theorem 4.1 and taking β to be constant hidden in the O of theorem 4.1 times a small factor to account for the factor $\frac{k+1}{k}$. □

In the remainder of the paper, β is assumed to be the constant implied in 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. Clearly, 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 elimination trees that is a factor of $O(\log^2 n)$ off optimal. Readers familiar with the approximation algorithms for constant treewidth of Lagergren [Lag90] and of Robertson and Seymour [RS86b] may note some similarities. Our algorithm also has some similarities to Lipton, Rose, and Tarjan's version of nested dissection [LRT79].

Our approximation algorithm consists of calling *makedec*$(V, \emptyset)$, where *makedec* is the following recursive procedure:

proc *makedec*(Z, W);
  (*comment: Z and W are disjoint sets of vertices.*)
  if $|Z \cup W| \leq 8\beta k \log n$ **then**
    return a tree-decomposition with one single node, containing $Z \cup W$.
  else *perform the following steps:*
    Find a 2/3 vertex separator S of W in $G[Z \cup W]$ with the algorithm of theorem 4.2.
    Find a 2/3 vertex separator S' of $Z \cup W$ in $G[Z \cup W]$, with the algorithm of theorem 4.2.
    Compute the connected components of $G[Z \cup W - (S \cup S')]$,
    suppose these have vertices $Z_i \cup W_i$ with $Z_i \subseteq Z$ and $W_i \subseteq W$
    for all $1 \leq i \leq t$.
    For $i := 1$ **to** t **do**
      call *makedec*$(Z_i, W_i \cup S \cup S')$.
    end for
    Now return the following tree-decomposition:
    take a root-node $r_{Z,W}$, containing $W \cup S \cup S'$ (i.e. $X_{r_{Z,W}} = W \cup S \cup S'$).
    Then add all tree-decompositions returned by the calls of
    makedec$(Z_i, W_i \cup S \cup S')$ with an edge from the root of each to $r_{Z,W}$.
  end if
end proc

Claim 5.1 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 $|Z \cup W| \leq 8\beta k \log n$. 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 $|W_i \cup S \cup S'| \leq 6\beta k \log n$. Clearly the first holds (use the assumption on the size of W, and use theorem 4.2 to bound the size of S and S'.) The second holds, as $S \cup S'$ 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 5.2 *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 3.1 and 3.2 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 5.3 *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 the tree-decomposition of G with the algorithm of theorem 5.2. Then use lemma 3.7. 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, and hence, by lemma 3.8, k is at most h. □

Similarly we can obtain:

Theorem 5.4 *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.*

6 Absolute approximations

In this section we consider absolute approximations, i.e. polynomial time algorithms that give solutions within an additive constant of the optimal solution. We show that when $P \neq NP$, then no such algorithms exist for the minimum height elimination tree problem, for treewidth (and hence for minimum front size and minmax clique), or for pathwidth.

Given an approximation algorithm $\mathcal{A}$ for a minimization problem we can distinguish between three kinds of performance guarantees on it. In *absolute approximations* the approximate solution $\mathcal{A}(I)$ is within a constant off the optimal solution $\mathcal{OPT}(I)$, i.e. $\mathcal{A}(I) - \mathcal{OPT}(I) \leq K$. Second, the approximate solution can be a constant factor off the optimal one, i.e., $\mathcal{A}(I) \leq C\,\mathcal{OPT}(I)$, with $C \geq 1$. Finally the difference between the optimal and approximate solutions can depend on the size of the problem, i.e. $\mathcal{A}(I) \leq f(n)\,\mathcal{OPT}(I)$. The algorithms we have presented above all have performance guarantees of the last kind with $f(n) = O(\log n)$ or $f(n) = O(\log^2 n)$. The hardest of these bounds to achieve is the absolute bound and very few NP-complete problems have absolute approximation algorithms. We will now prove that the minimum height elimination tree problem has no absolute approximation algorithms unless $P = NP$.

Theorem 6.1 *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 a fixed 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 the 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 ($V_1 \subseteq P$) and V_2 ($V_2 \subseteq Q$), with $|V_1| = |V_2| = k$, such that V_1 and V_2 are mutually independent?* That is: no edge joins a vertex in V_1 to a vertex in V_2. The MUS problem has been shown to be NP-complete [Pot88].

Let $B = (P, Q, E)$ be a bipartite graph. Its corresponding *biclique* $C = (P, Q, E \cup P^2 \cup Q^2)$ is the graph that contains enough extra edges to make each of P and Q into cliques. A bipartite graph is a *chain graph* if the adjacency sets of vertices in P form a chain, i.e., the vertices of P can be ordered such that

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

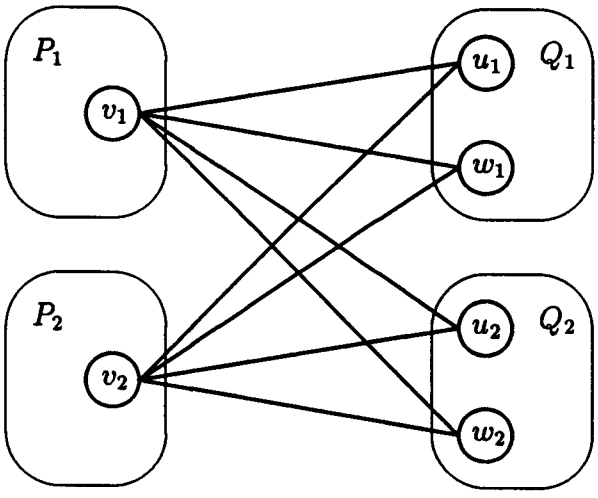

Figure 2: The graph $\hat{B}$ when $K = 1$

Yannakakis [Yan81] 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 [Pot88] has proved that B has mutually independent sets of size k iff there exists a chordal completion C' with elimination tree of height $n - k - 1$.

Suppose we had a polynomial time algorithm $\mathcal{A}$ for the minimum elimination tree problem, such that $\mathcal{A}(I) - \mathcal{OPT}(I) \leq K$. We solve MUS by making 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 2 we show $\hat{B}$ when $K = 1$.

Using the lemma from [Pot88] mentioned above we see that $\hat{B}$ has mutually independent sets of size $(K+1)k$ iff there exists a chordal completion $\hat{C}$ with an elimination tree of height $(K+1)(n-k)-1$. Assuming that we have an approximation algorithm $\mathcal{A}$ that gives us an elimination tree with height no more than K off the minimum we apply it to $\hat{C}$. If the resulting elimination tree has height between $(K+1)(n-k)-1$ and $(K+1)(n-k)+K-1$ then we can extract the $K+1$ elimination trees corresponding to the copies of the bipartite graph B that $\hat{B}$ was made up of. That gives us an elimination tree for C' of height $(n-k)-1$ (or lower). If the elimination tree computed by algorithm $\mathcal{A}$ is higher than $(K+1)(n-k)+K-1$ then the elimination tree corresponding to the original bipartite graph B (actually

its chordal completion C') cannot have height $(n-k)-1$. Otherwise we could join those together and obtain an elimination tree for $\hat{C}$ with height $(K+1)(n-k)-K-1$, which is at least $2K$ lower than the solution given by $\mathcal{A}$. Thus we could use the polynomial time algorithm $\mathcal{A}$ to solve an NP-complete problem. □

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

Lemma 6.2 *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 $W \subseteq X_i$.*

See [BM90] for a short proof of this lemma.

Theorem 6.3 *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, i.e. $V' = \{v_i \mid v \in V, 1 \leq i \leq K+1\}$, $E' = \{(v_i, w_j) \mid (v = w \wedge i \neq j) \vee (v, w) \in E\}$. We examine the relationship between the treewidth of G and the treewidth of G'.

Suppose we have a tree-decomposition of G, $(\{X_i \mid i \in I\}, T = (I, F))$ 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 $\text{treewidth}(G') \leq (\text{treewidth}(G)+1) \cdot (K+1) - 1$.

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 6.2 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 $\text{treewidth}(G') \geq (\text{treewidth}(G)+1) \cdot (K+1) - 1$, and hence that $\text{treewidth}(G') = (\text{treewidth}(G)+1) \cdot (K+1) - 1$.

Now we are able to describe the polynomial time algorithm for the treewidth problem: let G be the input graph. Make G', and apply algorithm $\mathcal{A}$ to G'. Apply the construction described above to make 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 $(k+1)(K+1)-1+K$, hence 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. With a different terminology this result was also proved by Deo et al. [DKL87].

Theorem 6.4 *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.*

7 Conclusion

We have presented a way to find bounded approximations to various parameters of graphs. To be precise: for treewidth, minimum front size and minimum maximum clique we obtain approximations that are never more than $O(\log n)$ times optimal, and for pathwidth and minimum height elimination tree we obtain approximations that are never more than $O(\log^2 n)$ times optimal.

An open problem is to find algorithms that give solutions that are only a constant times optimal for any of the parameters discussed in this paper. In this paper we have shown that there are no absolute approximations to the considered problems, but it is not known if we can find solutions that are only a constant factor off the optimal. Two related problems are how to permute A so that its Cholesky factor has the minimum fill or the minimum operation count. Klein et al. [KARR90] use a nested dissection algorithm somewhat similar to ours to give approximation algorithms for these measures getting within $O(\log^4 n)$ and $O(\log^6 n)$ times optimal (respectively) for graphs of bounded degree.

References

[ACP87] S. Arnborg, D. G. Corneil, A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8:277–284, 1987.

[ALS88] S. Arnborg, J. Lagergren, and D. Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Proc. 15 th ICALP*, pages 38–51. Springer Verlag, Lect. Notes in Comp. Sc. 317, 1988. To appear in J. of Algorithms.

[Arn85] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.

[Bod90a] H. L. Bodlaender. Polynomial algorithms for Graph Isomorphism and Chromatic Index on partial k-trees. *J. Algorithms*, 11:631–644, 1990.

[Bod90b] H. L. Bodlaender. Improved self-reduction algorithms for graphs with bounded treewidth. In *Proc. 15th Int. Workshop on Graph-theoretic Concepts in Computer Science WG'89*, pages 232–244. Springer Verlag, Lect. Notes in Computer Science, vol. 411, 1990. To appear in: Annals of Discrete Mathematics.

[BM90] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. In *Proc. 2nd Scandinavian Workshop on Algorithm Theory*, pages 301–309. Springer Verlag Lect. Notes in Computer Science vol. 447, 1990.

[DKL87] 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.

[DR83] 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.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

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

[Gil88] J. R. Gilbert. Some nested dissection order is nearly optimal. *Information Processing Letters*, 6:325-328, 1987/88.

[Har69] F. Harary. *Graph Theory*, Addison-Wesley, 1969.

[KARR90] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science*, IEEE, pages 726–737, 1990.

[Kön36] D. König. *Theorie der Graphen*, Reprinted by Chelsea Publishing Company, New York, 1950.

[Lag90] J. Lagergren. Efficient parallel algorithms for tree-decomposition and related problems. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science*, IEEE, pages 173–182, 1990.

[vL90] J. van Leeuwen. Graph Algorithms. In *Handbook of Theoretical Computer Science. A: Algorithms and Complexity Theory*, North Holland, Amsterdam, 1990.

[LR88] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximate algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, IEEE pages 422–431, 1988.

[Le90] F. T. Leighton. Personal communication, 1990.

[LL88] C. E. Leiserson and J. G. Lewis. Orderings for parallel sparse symmetric factorization. An unpublished manuscript, 1988.

[LRT79] R. J. Lipton, D. J. Rose and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis* 16:346-358, (1979).

[Liu90] J. W. H. Liu. The multifrontal method for sparse matrix solution: theory and practice. Tech. Report cs-90-04, York University, North York, Ontario, Canada M3J 1P3. To appear.

[Möh89] R. H. Möhring. Graph problems related to gate matrix layout and PLA folding. Technical Report 223/1989, Technical University of Berlin, 1989.

[Pot88] A. Pothen. The complexity of optimal elimination trees. Tech. Report CS-88-16, Pennsylvania State University, 1988.

[RS86a] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[RS86b] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. Manuscript, 1986.

[Yan81] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal of Algebraic and Discrete Methods*, 2:77–79, 1981.