A SEPARATOR THEOREM FOR CHORDAL GRAPHS

John R. Gilbert*

Donald J. Rose

TR 82-523

October 1982

Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853

*The work of this author was supported in part by National Science Foundation grant MCS-82-02948.

A Separator Theorem for Chordal Graphs

John R. Gilbert*
Computer Science Department
Cornell University
Ithaca, New York

Donald J. Rose
Bell Laboratories
Murray Hill, New Jersey

October 1982

Abstract

Chordal graphs are undirected graphs in which every cycle of length at least four has a chord. They are sometimes called rigid circuit graphs or perfect elimination graphs; the last name reflects their utility in modelling Gaussian elimination on sparse matrices. The main result of this paper is that a chordal graph with n vertices and m edges can be cut in half by removing $O(\sqrt{m})$ vertices. A similar result holds if the vertices have non-negative weights and we want to bisect the graph by weight, or even if we want to bisect the graph simultaneously by several unrelated sets of weights.

* The work of this author was supported in part by National Science Foundation grant MCS-82-02948.

I. Introduction

Many divide-and-conquer algorithms on graphs are based on finding a small set of vertices or edges whose removal divides the graph roughly in half. Examples include layout of circuits in a model of VLSI [Leis80], efficient sparse Gaussian elimination [Lipt79a, Gilb80], and construction of Voronoi diagrams to solve various geometric problems [Lipt80].

Most graphs do not have small separators that divide them evenly in half, but some useful ones do. Lipton and Tarjan's planar separator theorem gives an example.

Proposition. [Lipt79b] A planar graph with n vertices has a set of at most $2\sqrt{2n}$ vertices whose removal leaves no component with more than $2n/3$ vertices. □

This theorem is the best possible within a constant factor. Djidjev [Djid82] improved the constant $2\sqrt{2}$ to $\sqrt{6}$; the tightest possible constant is not known. Other kinds of graphs that can be separated evenly by deleting $o(n)$ vertices are trees ($O(1)$ vertices [Jord69, Lewi65]), outerplanar graphs ($O(1)$ vertices [Leis80]), hypercubes ($O(n/\sqrt{\log n})$ vertices [Gilb80]), graphs of genus at most g ($O(\sqrt{gn})$ vertices [Gilb82]), and several interconnection graphs for parallel computation [Hoey80, Leis80, Leig81].

An undirected graph is said to be *chordal* if every cycle of length at least four has a *chord*, which is an edge joining two vertices that are not adjacent on the cycle. Chordal graphs are perfect; that is, every induced subgraph of a chordal graph has a clique covering and an independent set of the same size [Hajn58]. Dirac [Dira61] developed some of the basic theory of chordal graphs, which he called rigid circuit graphs. Rose [Rose70] discovered a connection between chordal graphs and systems of linear equations whose coefficient matrices are sparse and symmetric. Such a system can be solved using Gaussian elimination with pivots chosen from the diagonal. The coefficient matrix is the adjacency matrix of an undirected graph; the graph is chordal if and only if the elimination can be done in some order without fill-in, that is, without changing any zero entries to non-zeroes.

Since a complete graph is chordal and has only trivial separators, chordal graphs in general cannot be separated by removing $o(n)$ vertices. The main result of this paper is that chordal graphs do satisfy a separator theorem in which the size of the separator depends on the density of the graph. We prove that a chordal graph with n vertices and m edges has a cutset of $O(\sqrt{m})$ vertices whose removal leaves no component with more than $n/2$ vertices. (This is immediate at the extremes of density, for complete graphs and for trees.) We show that the separator can in fact be chosen to be a complete subgraph. We also show that the result holds if the vertices have nonnegative weights and we want to bisect the graph by weight, or even if we want to bisect the graph simultaneously by several unrelated sets of vertex weights.

The next section contains some definitions and results from the literature that we will need later. Section III proves the main result. Section IV presents an almost linear algorithm to find the separator. Section V extends the main result to graphs whose vertices have multiple weights. The final section describes possible applications and open problems.

II. Results from the literature

The first results we require concern the graph model of Gaussian elimination. Let $G=(V,E)$ be a graph, not necessarily chordal. Let v be a vertex of G. The ***deficiency*** of v is the set of non-edges between neighbors of v,

$$D(v) = \{\{x,y\} : \{v,x\}\in E,\ \{v,y\}\in E,\ \{x,y\}\notin E\}.$$

The deficiency of v corresponds to the zeroes of the coefficient matrix that become nonzero when the equation in v's row is used to eliminate the variable in v's column. The graph G_v produced by ***eliminating*** v from G is obtained by adding v's deficiency and deleting v and its incident edges, so

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

When a sequence of vertices is eliminated from a graph, the edges in the deficiencies that are added are called ***fill-in*** edges. A ***simplicial vertex*** of a graph is a vertex that has a null deficiency, so it can be eliminated without fill-in; thus, it is a vertex whose neighbors form a clique. A graph G is a ***perfect elimination graph*** if its vertices can all be eliminated in some order without any fill-in. Such an order is called a ***perfect elimination ordering*** of the vertices of G.

The lemmas that follow are due to Rose and Tarjan [Rose70, Rose72, Rose76], Fulkerson and Gross [Fulk65], and Dirac [Dira61].

Lemma 1. A graph G is chordal if and only if it is a perfect elimination graph. □

A perfect elimination ordering must start with a simplicial vertex. Any simplicial vertex will do, and a choice of simplicial vertices is always available.

Lemma 2. If G is chordal and C is any clique then there is a simplicial vertex in $G-C$. Any simplicial vertex can be eliminated first in some perfect elimination ordering. □

Now we give a condition that determines the fill-in for any elimination ordering on any graph.

Lemma 3. Fix an elimination ordering for a graph G. Let v and w be nonadjacent vertices of G. Then $\{v,w\}$ is a fill-in edge if and only if there is a path from v to w consisting of vertices that are eliminated earlier than either v or w. □

A *separation clique* is a complete subgraph whose removal leaves a disconnected graph.

Lemma 4. If G is chordal and not complete, then G has at least one separation clique. □

We mention the following result to contrast it with the first theorem of the next section; the proofs below do not use it. A *v,w separator* is a set of vertices that cuts every path from v to w.

Lemma 5. A graph G is chordal if and only if for all vertices v and w, every minimal v,w separator in G is a clique. □

III. A $\sqrt{m}$-separator theorem

Let G be a chordal graph with n vertices and m edges. Suppose that each vertex of G has a nonnegative *weight*, and that the sum of the weights is n. The main result of this section is that there is a clique that divides the weight in half.

Theorem 1. Let G be a weighted chordal graph as above, with p vertices in its largest clique. Then G contains a clique whose removal leaves no connected component of weight more than $n/2$. Unless $n=1$, the clique can be chosen to have at most $p-1$ vertices.

Remark. This theorem resembles Lemma 5 above, but seems not to follow from it. Let us call the separator in the statement of Theorem 1 an *$n/2$ separator*. Then a minimal $n/2$ separator need not be a clique; for example, if G is a path with 5 vertices then one minimal $n/2$ separator is the second and fourth vertices. Also, there need not be a minimal v,w separator that is an $n/2$ separator; for example, if G is an $n/2$-vertex clique with an additional vertex of degree one adjacent to each clique vertex then the only minimal v,w separators are single clique vertices.

Proof. The idea of the proof is to start with an arbitrary clique and make it ooze around the graph like an amoeba until it is an $n/2$ separator. It oozes by disgorging vertices that can join or become components of weight less than $n/2$, and by engulfing vertices that are in a component of weight more than $n/2$.

Here are the details. We will not distinguish between a set of vertices of G and the subgraph of G it induces. Unless G is empty, it has at least one clique. Let C be the clique that minimizes the maximum weight of a connected component of $G-C$. In case of ties, minimize the number of vertices in a maximum-weight component of $G-C$. If ties remain, minimize the number of vertices in C. If ties still remain, choose arbitrarily.

Assume for the sake of contradiction that $G-C$ has a component A of weight greater than $n/2$. Then the total weight of $G-A$ is less than $n/2$. We shall state and prove three facts about A and C.

Fact 1. Every vertex of C is adjacent to some vertex of A.

Proof. If $v \in C$ were not adjacent to any vertex of A, then $C-\{v\}$ would have been chosen in preference to C.

Fact 2. If B is a nonempty subset of A, then B contains a vertex that is simplicial in $B \cup C$.

Proof. Immediate from Lemma 2.

Fact 3. Component A contains a vertex v adjacent to every vertex of C.

Proof. The vertex v is the last vertex of A in a perfect elimination ordering of $A \cup C$ with C ordered last. Thus $v = a_k$ where $\{a_1, \ldots, a_k, c_1, \ldots, c_h\}$ is a perfect elimination ordering of $A \cup C$. Such an ordering exists because by Fact 2 we can repeatedly choose simplicial vertices that are not in the clique C.

Let x be a vertex of C. Since A is connected and (by Fact 1) x is adjacent to a vertex of A, there is a path from x to a_k in $A \cup C$ that uses only vertices of $A-\{a_k\} = \{a_1, \ldots, a_{k-1}\}$. Lemma 3 says that if $\{x, a_k\}$ is not an edge of $A \cup C$ then it is a fill-in edge. But a perfect elimination ordering has no fill-in, so x is adjacent to a_k in $A \cup C$ and in G. Thus a_k is adjacent to every vertex of C, so we can take $v = a_k$.

Fact 3 leads to a contradiction: $C \cup \{v\}$ is a clique, and it should have been chosen in preference to C. Thus C is the desired $n/2$ separator.

The argument above shows that each component of $G-C$ contains a vertex adjacent to all of C's vertices. If G is not complete then C is not the largest clique in G, and C has at most $p-1$ vertices. If G is complete we can take C to be all of G except the lightest vertex. □

Corollary 1. Let G be a chordal graph with n vertices and m edges. Suppose G's vertices have nonnegative weights that add up to n. Then G has a set of $O(\sqrt{m})$ vertices whose removal leaves no connected component of weight more than $n/2$.

Proof. Theorem 1 says that G has a clique that separates the graph as required. This clique has at most m edges and hence only $O(\sqrt{m})$ vertices. □

Corollary 2. Let G be a chordal graph with n vertices and m edges. Then G has a set of $O(\sqrt{m})$ vertices whose removal leaves no connected component with more than $n/2$ vertices. □

A ***k-tree*** [Rose74] is a graph constructed by starting with a k-vertex clique and adding vertices one at a time, making each new vertex adjacent to k mutually adjacent old vertices. Thus a 1-tree is a tree. A k-tree is chordal, and its largest clique has $k+1$ vertices unless the k-tree is a k-clique. Therefore k-trees have $n/2$ separators whose size is independent of the size of the tree.

Corollary 3. Let T be a k-tree whose vertices have nonnegative weights that add up to n. Then T has a set of k vertices whose removal leaves no connected component of weight more than $n/2$. □

IV. An $O(m\,\alpha(m,n))$ algorithm

This section describes an algorithm to find the separator of Theorem 1 in $O(m\,\alpha(m,n))$ time, where $\alpha(m,n)$ is the very slowly growing function that appears in the running time of the disjoint set union algorithm [Tarj75].

We will develop the fast algorithm in several stages. The first stage comes directly from the proof of Theorem 1.

Algorithm 1.

```
begin
    $C \leftarrow \{\}$;
    while some component $A$ of $G-C$ has weight more than $n/2$ do
        while some vertex $x$ of $C$ is adjacent to no vertex of $A$ do
            $C \leftarrow C-\{x\}$
            od;
        $v \leftarrow$ some vertex of $A$ adjacent to every vertex of $C$;
        $C \leftarrow C \cup \{v\}$
        od
end
```

Since a vertex is added to C at most once, the main loop is executed at most n times. The whole algorithm is easily implemented to run in $O(mn)$ time.

We can speed this up by realizing that vertices are added to C in the reverse of a perfect elimination ordering of G.

Algorithm 2.

```
begin
    find a perfect elimination ordering $\{v_1,...,v_n\}$ of $G$;
    $i \leftarrow n+1$;
    while some component of $\{v_1,...,v_{i-1}\}$ has weight more than $n/2$ do
        $i \leftarrow i-1$
      od
    $C \leftarrow v_i$ plus all of $v_{i+1},...,v_n$ that are adjacent to $v_i$
end
```

We need to prove that this algorithm correctly finds the separator C of Theorem 1. Define the *boundary* of a set A of vertices, written ∂A, to be the set of vertices of $G-A$ that are adjacent to vertices of A. Consider the final value i takes on in the algorithm, and let A_i be the heaviest component of $\{v_1,...,v_i\}$.

First, C is a clique, because v_i is simplicial in $\{v_{i+1},...,v_n\}$ by the definition of a perfect elimination ordering.

Second, v_i must be in A_i, because A_i weighs more than $n/2$ but no component of $\{v_1,...,v_{i-1}\}$ weighs more than $n/2$. Then v_i is the last vertex of A_i in the perfect elimination ordering, so the same argument as the proof of Fact 3 in Theorem 1 shows that v_i is adjacent to every vertex in ∂A_i. Therefore C is $\partial A_i \cup \{v_i\}$. Since A_i is the only component of $G-\partial A_i$ that weighs more than $n/2$, $\partial A_i \cup \{v_i\}$ is a set whose deletion from G leaves no component that weighs more than $n/2$.

Algorithm 2 also runs in $O(mn)$ time. We can get a more efficient implementation by looking at the vertices in the opposite order, from v_1 to v_n. The final version of the algorithm starts with an empty graph H and adds vertices of G to it in perfect elimination order until some component of H has weight more than $n/2$. The last vertex this algorithm adds to H is vertex v_i from Algorithm 2; knowing v_i we can easily find C.

The following algorithm maintains a family of disjoint sets $s_1,\ldots,s_n$ of vertices to represent the components of H. Initially all the sets are empty. As vertex v_i comes up in perfect elimination order it is placed in set s_i, and then the sets containing vertices adjacent to v_i are merged into s_i. Along with each set s_i we keep the total weight $w(s_i)$ of its vertices. Function find(v) returns the name of the set containing vertex v.

Algorithm 3.

```
begin
    find a perfect elimination ordering $\{v_1,\ldots,v_n\}$ of $G$;
    $i \leftarrow 0$;
    repeat
        $i \leftarrow i+1$;
        $s_i \leftarrow \{v_i\}$;
        $w(s_i) \leftarrow$ weight of $v_i$;
        for $v_j$ adjacent to $v_i$ with $j<i$ do
            $s \leftarrow$ find($v_j$);
            if $s \neq s_i$ then
                $s_i \leftarrow s_i \cup s$;
                $w(s_i) \leftarrow w(s_i) + w(s)$
            fi
        od
    until $w(s_i) > n/2$;
    $C \leftarrow v_i$ plus all of $v_{i+1},\ldots,v_n$ adjacent to $v_i$
end
```

The analysis of this algorithm's running time is straightforward. Finding a perfect elimination ordering takes $O(m)$ time by an algorithm of Rose, Tarjan, and Leuker [Rose76]. The main repeat loop is executed at most n times. The for loop is executed at most twice for each edge, and its body requires constant time except for the find operation. The time for at most $2m$ finds is $O(m\alpha(m,n))$. Finally, the computation of C takes $O(m)$ time. Thus the total running time is dominated by the finds, and is $O(m\alpha(m,n))$.

V. More separator theorems

We can use the separator theorem in Section III to find separators that chop a chordal graph into fragments no larger than a specified weight, or to separate a chordal graph according to two or more unrelated sets of weights simultaneously. These are analogous to results of Lipton, Tarjan, and Gilbert on graphs with $\sqrt{n}$-separators, as referenced below.

Theorem 2. [Lipt80, Theorem 2] Let G be a chordal graph with n vertices and m edges, with nonnegative vertex weights that add up to 1. Let $\epsilon>0$ be given. Then there is a set of $\sqrt{m/\epsilon}$ vertices of G whose removal leaves no component with weight more than ϵ. The separator can be found in $O(m \log n \, \alpha(m,n))$ time. □

Theorem 3. [Gilb80, Theorem 1.3.2] Let G be a chordal graph with n vertices and m edges, with two sets of vertex weights that both add up to n. Then the vertices of G can be partitioned into sets A, B, and C such that C separates A from B, C has $O(\sqrt{m})$ vertices, neither A nor B has weight of the first kind more than $n/2$, and neither A nor B has weight of the second kind more than $(1/2+\epsilon)n$. The separator can be found in $O(m\alpha(m,n))$ time. □

For example, we can take one kind of weight to be 1 for every vertex, and the other kind to be the vertex degree. Then Theorem 3 allows us to divide a chordal graph into two pieces with equal numbers of vertices and roughly equal numbers of edges.

Theorem 3 can be applied recursively to obtain a $\sqrt{m}$-separator theorem for any fixed number of sets of vertex weights. In general the separator divides the graph into two pieces, each with at most half the first kind of weight and at most $1/2+\epsilon$ of each of the other kinds of weight. This quickly ceases to be practical, because the constant factor in the proof of the theorem grows double-exponentially with the number of kinds of weight.

VI. Remarks

The algorithm in Section IV finds a separator in $O(m\alpha(m,n))$ time. We suspect that this can be improved to $O(m)$ time. No algorithm faster than $O(m)$ is possible if the input graph is represented by listing its edges or by listing the vertices adjacent to each vertex. However, the graph might be more compactly represented. For example, the chordal graph corresponding to an acyclic hypergraph (as described below) is given as a union of cliques. The sum of the sizes of the cliques can be much less than m. It might be possible to find a separator in time nearly linear in this sum.

Chordal graphs have applications in a number of areas, and we expect the separator theorems above to be useful in some of them. One such area is solving sparse linear systems by Gaussian elimination, where one wishes to find an elimination ordering for a sparse matrix that causes relatively few zeroes to become nonzero. If the matrix is symmetric and only symmetric permutations are allowed, this corresponds to finding a small set of edges whose addition makes a graph chordal. Finding the smallest such set of edges is an NP-complete problem [Yann81], but there are heuristics that perform well in many cases. One heuristic, called nested dissection, uses separators in planar graphs to give good orderings for systems that come from differential equations on two-dimensional regions [Lipt79a]. We are investigating the use of the chordal separator theorem to show that any graph has some nested dissection ordering that is close to optimum.

Chordal graphs and their separators also appear in database theory. A database represents a relation (called a *universal relation*) on a set of attributes. The universal relation can be represented implicitly by explicitly storing its projections on some subsets of the attributes. Consider the hypergraph whose vertices are attributes and whose hyperedges are the subsets whose projections are explicitly stored. A separator in this hypergraph implies an association (technically, a *multivalued dependency*) between each component and the separator. This association often corresponds to some relationship in the real world among the attributes involved.

The hypergraphs of database schemes from the real world are nearly all *acyclic* [Beer81]. Acyclic database schemes have a number of desirable properties; roughly, in an acyclic scheme pairwise consistency between the projections implies that the universal relation is consistent. A hypergraph is acyclic if and only if the graph formed by replacing each hyperedge with a clique is chordal.

Acknowledgement. We thank Dale Skeen for several stimulating discussions, and particularly for helping us understand acyclic database schemes.

References

[Beer81]
Catriel Beeri, Ronald Fagin, David Maier, Alberto Mendelzon, Jeffrey Ullman, and Mihalis Yannakakis.
Properties of acyclic database schemes.
Proceedings of the 13th Annual ACM Symposium on Theory of Computing, pages 355-362, 1981.

[Dira61]
G. A. Dirac.
On rigid circuit graphs.
Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg **25**: 71-76, 1961.

[Djid82]
Hristo Nicolov Djidjev.
On the problem of partitioning planar graphs.
SIAM Journal on Algebraic and Discrete Methods **3**: 229-240, 1982.

[Fulk65]
D. R. Fulkerson and O. A. Gross.
Incidence matrices and interval graphs.
Pacific Journal of Mathematics **15**: 835-855, 1965.

[Gilb80]
John Russell Gilbert.
Graph Separator Theorems and Sparse Gaussian Elimination.
Ph. D. Thesis, Stanford University, 1980.

[Gilb82]
John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan.
A separator theorem for graphs of bounded genus.
Cornell University Department of Computer Science technical report 82-506, 1982.

[Hajn58]
András Hajnal and János Surányi.
Über die Auflösung von Graphen in vollständige Teilgraphen.
Annales Universitatis Scientarium Budapest, Sectio Mathematica **1**: 113-121, 1958.

[Jord69]
Camille Jordan.
Sur les assemblages de lignes.
Journal für die reine und angewandte Mathematik **70**: 185-190, 1869.

[Hoey80]
Dan Hoey and Charles E. Leiserson.
A layout for the shuffle-exchange network.
Carnegie-Mellon University Department of Computer Science
technical report CMU-CS-80-139, 1980.

[Leig81]
Frank Thomson Leighton.
New lower bound techniques for VLSI.
Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, pages 1-12, 1981.

[Leis80]
Charles E. Leiserson.
Area-efficient graph layouts (for VLSI).
Proceedings of the 21st Annual Symposium on Foundations of Computer Science, pages 270-281, 1980.

[Lewi65]
P. M. Lewis II, R. E. Stearns, and J. Hartmanis.
Memory bounds for the recognition of context-free and context-sensitive languages.
IEEE Conference Record on Switching Theory and Logical Design,
pages 191-202, 1965.

[Lipt79a]
Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan.
Generalized nested dissection.
SIAM Journal on Numerical Analysis **16**: 346-358, 1979.

[Lipt80]
Richard J. Lipton and Robert Endre Tarjan.
Applications of a planar separator theorem.
SIAM Journal on Computing **9**: 615-627, 1980.

[Lipt79b]
Richard J. Lipton and Robert Endre Tarjan.
A separator theorem for planar graphs.
SIAM Journal on Applied Mathematics **36**: 177-189, 1979.

[Rose72]
Donald J. Rose.
A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations.
In *Graph Theory and Computing* (Ronald C. Read, editor), pages 183-217, Academic Press, 1972.

[Rose74]
Donald J. Rose.
On simple characterizations of k-trees.
Discrete Mathematics **7**: 317-322, 1974.

[Rose70]
Donald J. Rose.
Triangulated graphs and the elimination process.
Journal of Mathematical Analysis and Applications **32**: 597-609, 1970.

[Rose76]
Donald J. Rose, R. Endre Tarjan, and George S. Leuker.
Algorithmic aspects of vertex elimination on graphs.
SIAM Journal on Computing **5**: 266-283, 1976.

[Tarj75]
Robert Endre Tarjan.
Efficiency of a good but not linear set union algorithm.
Journal of the ACM **22**: 215-225, 1975.

[Yann81]
Mihalis Yannakakis.
Computing the minimum fill-in is NP-complete.
SIAM Journal on Algebraic and Discrete Methods **2**: 77-79, 1981.