Angewandte Mathematik und
Informatik
Universität zu Köln

Report No. 355

Minimum Fill-in and Treewidth for Graphs Modularly Decomposable into Chordal Graphs

by
Elias Dahlhaus

1999

A preliminary version appeared in WG98 [7], partially supported by ESPRIT Long Term Research Project Nr. 20244 (ALCOM-IT)

Institut für Informatik, Universität zu Köln,
Pohligstrasse 1, 50969 Köln, Germany

Abstract

We show that a minimum fill-in ordering of a graph can be determined in linear time if it can be modularly decomposed into chordal graphs. This generalizes results of [2]. We show that the treewidth of these graphs can be determined in $O((n+m)\log n)$ time.

1 Introduction

One of the major problems in computational linear algebra is that of sparse Gauss elimination. The problem is to find a pivoting, such that the number of zero entries of the original matrix that become non zero entries in the elimination process is minimized. In case of symmetric matrices, we would like to restrict pivoting along the diagonal. The problem translates to the following graph theory problem [14].

Minimum Elimination Ordering: For an ordering $<$ on the vertices, we consider the fill-in graph $G'_< = (V, E')$ of $G = (V, E)$. $G'_<$ contains first the edges in E and secondly two vertices x and y form an edge in $G'_<$ if they have a common smaller neighbor in $G'_<$. *The problem of Minimum Elimination ordering is, given a graph $G = (V, E)$, find an ordering $<$, such that $G'_<$ has a minimum number of fill-in edges.*

Note that this problem is NP-complete [16] in general. There is a polynomial time solution for this problem for so called HHD-free graphs [4]. Moreover, for distance hereditary graphs and for certain graph classes with few P4-s, there are linear time solutions [3, 2]. Here we generalize the result of [2] and show that a minimum fill-in ordering can be determined in linear time if the graph can be modularly decomposed into chordal graphs.

Another problem is to find an elimination scheme, such that the size of "dense matrices" is as small as possible. This is related to the problem of treewidth.

Treewidth: Find an ordering $<$, such that the maximum clique size of $G'_<$ is minimized.

Also the problem of minimum treewidth is NP-complete [1]. The problem has a polynomial time solution for HHD-free graphs [4].

We will show that we can compute a minimum elimination ordering and the treewidth for graphs that can be modularly decomposed into chordal graphs in linear time or almost in linear time.

In section 2, we introduce the notation of the paper. Section 3 presents a linear time algorithm for minimum fill-in for graphs that can be modularly decomposed into chordal graphs. Section 4 discusses the treewidth of graphs that can be modularly decomposed into chordal graphs.

2 Notation

A *graph* $G = (V, E)$ consists of a *vertex set* V and an *edge set* E. Multiple edges and loops are not allowed. The edge joining x and y is denoted by xy.

We say that x is a *neighbor* of y iff $xy \in E$. The set of neighbors of x is denoted by $N(x)$ and is called the *neighborhood*. Analogously, for a set X of vertices, $N(X)$ is the set of neighbors of some vertex in X that are not in X and $N[X]$ is the set of neighbors of vertices in X together with the vertices in X.

Trees are always directed to the root. The notion of the *parent*, *child*, *ancestor*, and *descendent* are defined as usual.

A *subgraph* of (V, E) is a graph (V', E') such that $V' \subseteq V$, $E' \subseteq E$. The graph $G[X]$ is the subgraph *induced* by X consisting of all vertices in X and all edges $xy \in E$ with $x, y \in E$.

We denote by n the number of vertices and by m the number of edges of G.

A graph is called *chordal* iff each cycle of length greater than three has a chord, i.e. an edge that joins two nonconsecutive vertices of the cycle. Note that chordal graphs are exactly those graphs having a *perfect elimination ordering* $<$, i.e. for each vertex v the neighbors $w > v$ induce a complete subgraph, i.e. they are pairwise joined by an edge [9].

Note that in any chordal graph, the number of maximal cliques is bounded by n and the number of pairs (x, c) such that x is in the clique c is bounded by $n + m$.

The *fill-in* of a graph $G = (V, E)$ and an ordering $<$ is the smallest edge set E', such that $E \subseteq E'$ and $<$ is a perfect elimination ordering of $G_< := (V, E')$. Note that $G_<$ is chordal. The problem to get a minimum fill-in or a minimum elimination ordering is to get an ordering $<$, such that the fill-in is minimum.

The *treewidth* of $G = (V, E)$ is the minimum maximum clique size of a chordal graph $G' = (V, E')$ with $E \subseteq E'$.

By a *module* of a graph $G = (V, E)$, we define a subset V' of the vertex set V such that all vertices in V' have the same neighbors outside V'. We do not compute all modules but those modules X which do not *overlap* with other modules, i.e. there is no module Y that has a nonempty intersection with X and neither $X \subset Y$ nor $Y \subset X$. We call such modules also *overlap free*. Note that the overlap-free modules of any graph form a tree with respect to set inclusion, i.e. two overlap free modules are disjoint or comparable with respect to set inclusion. The notions of parent and child modules can be defined in an obvious way. The parent $P(X)$ of an overlap-free module X is the unique smallest overlap free module that is a proper superset of X. Y is a child module of X if and only if Y is an inclusion maximal proper overlap free submodule of X. The system of overlap-free modules is also called the *modular decomposition* of the graph G.

Note that a modular decomposition can be determined in linear time [12, 6, 8].

Let G be a graph with modules $V_1, \ldots, V_k$. Then $G/(V_1, \ldots, V_k)$ is the graph we obtain from G by shrinking each V_i to one vertex. Let X be a module with child modules $Y_1, \ldots, Y_k$. $G_X = G[X]/(Y_1, \ldots, Y_k)$ is the graph we get from $G[X]$ by shrinking each child module of Y_j of X to one vertex.

We call a graph *modularly decomposable into chordal graphs* or shortly *modulated chordal* if for all overlap-free modules X of G, G_X is chordal.

3 Minimum Fill-in of a Graph with a Modular Decomposition into Chordal Graphs

We follow the ideas of [2]. The key lemma is the following.

Lemma 1 *Let V' be a module of G and let $N(V')$ be the set of neighbors of V' that do not belong to V'. Then in any fill-in E' of G, V' is complete or $N(V')$ is complete.*

Proof: Assume V' and $N(V')$ are both not complete in E', i.e. there are $u, v \in V'$ and $u', v' \in N(V')$ that are not joint by an edge in E'. Then they form a cycle of length four in G and also in E'.

Q.E.D.

Corollary 1 *Suppose V_1 and V_2 are disjoint modules of G and all vertices in V_1 are adjacent with all vertices in V_2. Then in any fill-in E' of G, V_1 or V_2 is a complete set.*

We immediately get the following.

Corollary 2 *Let $V_1, \ldots, V_k$ be the child modules of G. Then for any fill-in E', the set of V_i that are not complete in E' form an independent set in*

$$G/(V_1, \ldots, V_k).$$

Now we assume that $V_1, \ldots, V_k$ are the child modules of G and

$$G' = G/(V_1, \ldots, V_k)$$

is a chordal graph.

To get a minimum fill-in, we first recursively compute a minimum fill-in E'_i, for all V_i. Then we select an independent set V' of vertices of G' as those modules V_i that are not made complete. The neighborhoods of the modules in V' are made complete. The modules corresponding to vertices of G' not belonging to V' are made complete. The number of resulting fill-in edges has to be minimized.

The following result proves that the resulting graph is a chordal graph and is easy to check.

Lemma 2 *1. Let G be a chordal graph and v be a vertex of G. Then the graph that comes up by replacing v by a complete set V_1 with the same neighbors outside V_1 as v is a chordal graph.*

2. Let G be a chordal graph and v be a vertex of G. Then the graph G' that comes up by making the neighborhood of v complete is a chordal graph.

3. Let G be a chordal graph and v be a simplicial vertex (i.e. the neighborhood is complete) of G. Then the graph that comes up by replacing v by a module that is a chordal graph is chordal.

Proof: We always can assume that G has a perfect elimination ordering $<$. When we replcace v by consecutive pairwise adjacent vertices $v_1, \ldots, v_k$, i.e. $v_1 < v_2 < \ldots < v_k$, $w < v$ iff $w < v_1$, and $v < w$ iff $v_k < w$, for each vertex $w \neq v$ of G, $<$ remains a perfect elimination ordering. This proves the first statement of the lemma.

The second part is proved as follows. We show that $<$ remains a perfect elimination ordering. Let $w < w_1, w_2$ and $ww_\nu \in E$ or w and w_μ are both neighbors of v. It has to be shown that $w_1w_2 \in E$ or that w_1 and w_2 are both neighbors of v. If w is not a neighbor of v then $w_1w_2 \in E$. If w is a neighbor of v and $v \leq w$ then $ww_1 \in E$ and $ww_2 \in E$, because they are neighbors of w or greater neighbors of v (and therefore adjacent to the greater neighbor w of v). Therefore also $w_1w_2 \in E$. Finally let $w < v$ be a neighbor of v. If w_1 or w_2 is a neighbor of w then it is also a neighbor of v (v and w_1 or w_2 arte greater neighbors of w). In any case, w_1 and w_2 belong to the neighborhood of v.

The third part is proved as follows. We always have a perfect elimination ordering $<$ of G that starts with the simplicial vertex v. Let M be another chordal graph with a perfect elimination ordering $<'$. When we replace v by M as a module then the concatenation of $<'$ and $<$ restricted to $G - v$ is a perfect elimination ordering.

Q.E.D.

To get the right independent set V', we proceed as follows. For each module V_i, let f_i be the number of fill-in edges one gets if V_i is made complete, i.e. the number of non edges in V_i, and let g_i be the number of fill-in edges one gets if V_i is not made complete, i.e. the number of fill-in edges of a minimum fill-in of $G[V_i]$ plus the number of non edges in the neighborhood of V_i that join vertices that appear in different V_j.

Lemma 3 *The number of fill-in edges that are created by making exactly the modules in V' not complete (and making the remaining modules complete) is*

$$\Sigma_{V_i \in V'} g_i + \Sigma_{V_i \notin V'} f_i.$$

Proof: Note that V' is an independent set in $G' = G/(V_1, \ldots, V_k)$ and that fill-in edges might be created by two modules V_i and V_j only in case that they

are common non edges of the neighborhoods of V_i and V_j. This can only be the case if V_i and V_j belong to V'. Now V_i and V_j are not joint by an edge in G'. Since G' is a chordal graph, all vertices in the joint neighborhood of V_i and V_j are pairwise joint by an edge (otherwise G' had a chordless cycle of length four. Therefore no fill-in edge V_pV_q is created by two V_i's in V'. This proves the lemma.

Q.E.D.

To get the size of a minimum fill-in of G, one has to compute a maximum weighted independent set of $G' = G/(V_1, \ldots, V_k)$, where the weight of V_i is $f_i - g_i$.

A. Frank [10] stated an algorithm to determine a maximum weighted independent set in a chordal graph. He proved that the algorithm has a polynomial time bound. The algorithm has in fact a linear time bound.

Lemma 4 *[10] We can determine a maximum weighted independent set in a chordal graph G' in linear time.*

If the numbers of vertices of V_i and edges of V_i are known, one gets f_i immediately. To get the number of edges of V_i, one first determines, for each edge e of G, the smallest overlap-free module X_e that contains e (by determining the least common ancestor). Then we determine the number $ed(X)$ of e with $X_e = X$, for each overlap-free module X. To get the number of edges of any overlap-free module, we recursively add the number of edges in each child module of X and $ed(X)$.

To get g_i, one has to compute the number of non edges in the neighborhood of V_i that are not in the same V_j. We have the number of non edges of the neighborhood of V_i if we have the number of edges in the neighborhood of V_i that are not in the same V_j. We consider the chordal graph $G/(V_1, \ldots, V_k)$ and weight each edge V_iV_j by $|V_i||V_j|$.

Lemma 5 *For a chordal graph G' with vertex weights $w(v)$, for each vertex v and edge weights $w(e) = w(vw) = w(v)w(w)$, for each edge $e = vw$ of G', we can compute, for all vertices v of G' simultaneously, the sum of edge weights in the neighborhood of v in G' in linear time.*

Proof: We assume that a perfect elimination ordering of G' is known. Let $h(v)$ be the sum of edge weights of edges that join neighbors x and y of v that are greater than v. Since greater neighbors of v are pairwise adjacent,

$$h(v) = \Sigma_{x,y>v,xv,yv\in E,x\neq y} w(x)w(y).$$

This can be replaced by

$$h(v) = ((\Sigma_{xv\in E,x>v} w(x))^2 - \Sigma_{xv\in E} w(x)^2)/2.$$

Therefore all $h(v)$ can be determined in linear time.

Next we have to consider neighbors x and y, such that at least one of x or y is smaller than v. Without loss of generality, $x < y$ and $x < v$. Note that if y is a neighbor of x then y is a neighbor of v.

Let $w'(x)$ be the sum of edge weights $w(xy)$ with $xy \in E$ and $x < y$. Note that all $w'(x)$ can be determined simultaneously in linear time.

The sum of all edge weights in the neighborhood of v is determined by

$$s(v) := h(v) + \Sigma_{x<v, xv \in E}(w'(x) - w(xv)).$$

Q.E.D.

Corollary 3 *g_i, $i = 1, \ldots, k$ can be determined in linear time.*

Proof: By previous lemma, we know the sum $s(V_i)$ of weights $w(V_p, V_q) = |V_p||V_q|$ of neighbors V_p and V_q of V_i, such that V_p and V_q are adjacent in $G' = G/(V_1, \ldots, V_k)$. Let $N(V_i)$ be the neighborhood of V_i in G'. Then the sum of $|V_p||V_q|$, $p \neq q$, $V_p, V_q \in N(V_i)$ is

$$t_i = ((\Sigma_{V_q \in N(V_i)}|V_q|)^2 - \Sigma_{V_q \in N(V_i)}|V_q|^2)/2.$$

t_i can be determined in linear time and $g_i = h_i + t_i - s(V_i)$ where h_i is the size of a minimum fill-in of $G[V_i]$. Therefore g_i can be determined in linear time.

Q.E.D.

Theorem 1 *The size of a minimum fill-in of a modulated chordal graph can be determined in linear time.*

Proof: One determines the sizes of the minimum fill-ins of all overlap-free modules X recursively. Note that one recursion step can be done in linear time with respect to the size of G_X. Therefore the overall time is linear with respect to the the size of the whole graph G.

Q.E.D.

It remains to get a minimum fill-in ordering i.e. a perfect elimination ordering of the fill-in. Note that we did not compute the edges of the fill-in graph explicitly. We follow the proof of Lemma 2. We may assume that for all modules of G, a perfect elimination ordering is known.

We proceed again recursively.

1. We assume that we know minimum fill-in orderings $<_i$ of $G[V_i]$ and we know the set V' of those V_i that are not made complete in a minimum fill-in of G. We also assume that a perfect elimination ordering $<'$ of $G' = G/(V_1, \ldots, V_k)$ is known.

2. We first concatenate the orderings $<_i$, such that $V_i \in V'$ and get an ordering $<'_1$ of the vertices of V that appear in some $V_i \in V'$. Then we concatenate the orderings $<_i$ with $V_i \notin V'$, such that $<_i$ appears before $<_j$ if $V_i <' V_j$ in the perfect elimination ordering of G'. We get an ordering $<'_2$.

3. The final elimination ordering $<$ of the fill-in graph is the concatenation of $<'_1$ first and $<'_2$ second.

Lemma 6 *The ordering $<$ as constructed above is a perfect elimination oprdering of the minimum fill-in graph.*

Proof: This follows from the proof of lemma 2.

1. Assume $V_i \in V'$. Then the neighborhood of V_i is made complete. $<'$ remains a perfect elimination ordering of G' if we make the neighborhood of V_i complete. If we replace V_i by the fill-in graph of $G[V_i]$ then we can take $<_i$ first and $<'$ second and we have a perfect elimination ordering of of the graph G'_i that comes up when we replace V_i by the minimum fill-in graph of $G[V_i]$.

2. Since V' is independent in G', we replace all $V_i \in V'$ by the minimum fill-in graph of $G[V_i]$ and the concatenation of the $<_i$ with $V_i \in V'$ first and $<'$ restricted to the $V_i \notin V'$ second is a perfect elimination ordering of the graph $G'_{V'}$ that comes up if we replace each V_i by the minimum fill-in graph of $G[V_i]$.

3. We can replace each vertex V_i of $G'_{V'}$ by complete set D_i and the graph remains chordal. In the perfect elimination ordering $<'_{V'}$ we may replace V_i by the consecutive enumeration of the vertices in D_i, and we still have a perfect elimination ordering. This is in particular true if we replace V_i as a vertex of $G'V'$ by the vertices of V_i.

At the end we get the ordering as constructed above.

Q.E.D.

It remains to show that the ordering as constructed above can be determined in linear time. We may assume that the perfect elimination orderings of the modules are given by lists and not by enumerations. Then in each recursion step, the minimum fill-in ordering $<$ can be determined from the orderings $<_i$ in $O(n)$ time, where n is the number of V_i. As a final result, we get the following.

Theorem 2 *A minimum fill-in ordering of a modulated chordal graph can be determined in linear time.*

4 Treewidth of Modulated Chordal Graphs

We will show the following.

Theorem 3 *The treewidth of a modulated chordal graph can be determined in $O(n+m)\log n$ time.*

Proof: We proceed in a similar way as in the case of minimum fill-in. We recursively determine the treewidths of the maximal modules $V_1, \ldots, V_k$ and select an appropriate independent set I of $G/(V_1, \ldots, V_k)$, such that the $V_i \in I$ are exactly those modules that are not made complete. We may assume that a perfect elimination ordering of $G/(V_1, \ldots, V_k)$ is known. We assume that just the enumeration $V_1, \ldots, V_k$ defines a perfect elimination ordering of $G/(V_1, \ldots, V_k)$. In any fill-in G' of G, there are two kinds of cliques.

1. Cliques c that are unions of V_i, i.e. there is a V_i, such that $c = V_i \cup \bigcup_{V_j > V_i, V_i V_j \in E} V_j$. Note that in this case, all V_j in c are made complete.
2. Cliques c that are not unions of V_i, i.e. there is a clique c_1 of $G'[V_i]$, such that $c = c_1 \cup N(V_i)$.

We assume that we know the treewidth t_i of $G[V_i]$. Let $s_i^1 := t_i + |N(V_i)|$ and $s_i^2 := |V_i| + \Sigma_{V_j > V_i, V_i V_j \in E} |V_j|$. s_i^1 is the maximum size of a clique that intersects V_i if V_i is not made complete. s_i^2 is the size of V_i together with its greater neighborhood. For an independent set I of $G/(V_1, \ldots, V_k)$, let J be the set of $V_i \notin I$, such that all modules V_j in the greater neighborhood of V_i are not in I. The each clique that is a union of V_j is of the size s_i^2, for some $i \in J$ and each clique that is not the union of some V_j is of the size s_i^1, for some $i \in I$. The maximum clique size of the fill-in G_I associated with I is denoted by S_I and can be determined as follows.

$$S_I := \max(s_i^1 : V_i \in I, s_i^2 : V_i \in J).$$

The treewidth is therefore determined by

$$S := \min_I S_I.$$

We again consider the elimination tree T with parent function Par where $Par(V_i)$ is the next greater neighbor of V_i in $G/(V_1, \ldots, V_k)$. Let D_i be the set of descendents of V_i including V_i in T. For an independent set I_i of D_i, let J_i be the set of $V_j \notin I$ in D_i, such that no greater neighbor of V_j is in I. Then $S_{I_i}^i := \max(s_j^1 | V_j \in D_i \cap I_i, s_j^2 | V_j \in J_i)$ and $S^i = \min_{I_i} S_{I_i}^i$.

To get S, we determine the S^i recursively. Let $S_i^1 := \min_{V_i \in I_i} S_{I_i}^i$ and $S_i^2 := \min_{V_i \notin I_i} S_{I_i}^i$. Then

$$S_i^1 = \max(s_i^1, (S^j : V_j \in D_i, V_j V_i \notin E, Par(V_j) V_i \in E))$$

and

$$S_i^2 = \max(s_i^2, (S^j : Par(V_j) = V_i)).$$

Note that $S^i = \min(S_i^1, S_i^2)$.

Obviously, the time bound is $O(n^2)$. But we also can get $O(n+m) \log n$ as follows. We sort the children V_j of each V_i by the numbers S^j as soon as we

know the S^j, for all children V_j of V_i. This takes $O(n+m)+O(n\log n)$ time. For each descendent V_j of V_i (i.e. $V_j < V_i$) that is a neighbor of V_i, we now can determine a child V_l of V_j that is not a neighbor of V_i of maximum S^l in the order of the number of children of V_j that are neighbors of V_i. This gives a time bound in the order of the number of neighbors of V_i times a logarithmic factor. The overall complexity of the algorithm is therefore $O(n+m)\log n$.

Q.E.D.

Remark 1 *The additional logarithmic factor comes from the fact that we have to sort the children of any node of the elimination tree. It might be interesting to improve the algorithm in such a way that we can circumvent sorting.*

5 Introducing Modules of Bounded Size

We now consider the case that G can be modularly decomposed into prime graphs that are chordal *or of bounded size*. To find a minimum fill-in ordering of G, we had to know the minimum fill-in ordering of all maximal modules V_i of G. It was not essential that the modules V_i themselves again induce modulated chordal graphs. It only was essential that we can get the minimum fill-in orderings of the V_i in linear time. It remains therefore to discuss the case that $G' = G/(V_1,\ldots,V_k)$ is a graph of bounded size. Let again V' be the set of V_i that do not induce a complete subgraph of the fill-in. *Then V' is an independent set of G'.* The proof is the same as the proof of lemma 1. The vertices of V_i and V_j not in V' that have a common neighbor in V' must be joined by fill-in edges. Therefore given V', we make all V_i and V_j adjacent, i.e. having an edge between every vertex of V_i and every vertex of V_j, if they are adjacent to a common $V_k \in V'$. We call these edges V'-fillin edges. It remains to determine a minimum fill-in of the graph $G'' = G''_{V'}$ that arises from $G' - V'$ together with the V'-fill-in edges, where each V_i is replaced by a clique of size $|V_i|$. Define a set S of the vertices of G' to be a cut if $G'' - S$ has at least two connected components, say C_1 and C_2, such that all vertices of S are in the neighborhood of C_1 and of C_2. By a result of Parra and Scheffler [13], all cuts of a minimum fill-in (even of a fill-in that is minimal with respect to the subset relation) of G'' are also cuts of G'' and all fill-in edges join two vertices that are in a common cut of the minimum fill-in. It is easy to observe that in all cuts S of G'', either all vertices of any particular V_i or none of its vertices belong to S. *Therefore either all vertices of V_i and V_j are pairwise joined by a fill-in edge or none of them.* We again weight a fill-in edge V_iV_j with $w(V_i,V_j) = |V_i||V_j|$ and each V_i with the number f_i of its non edges and the minimum fill-in size g_i of the minimum fill-in of $G[V_i]$. Knowing the sizes of the minimum fill-ins of the V_i's, we can get the size of a minimum fill-in of G in constant time. We only have go through all possible V' and all fill-ins of $G''_{V'}$. We weight the $V_i \in V'$ with g_i, the vertices not in V' with f_i, and the fill-in edges with its weight $w(V_i,V_j)$. We select the $G''_{V'}$ with the smallest weight sum.

Theorem 4 *The size of a minimum fill-in of a graph whose prime modules are chordal or of bounded size can be determined in linear time.*

To get the minimum treewidth, we can proceed in the same way and get a time bound of $O(n+m)\log n$.

This generalizes a result of [2] that in "graphs with few P_4-s, the size of a minimum fill-in and the minimum treewidth can be determined in linear time.

6 Conclusions

It might also be possible to extend the ideas also to HHD-free graphs. One should mention that in HHD-free graphs, there is always a vertex, such that the in the neighborhood, the connected components of the complement form modules [11]. A polynomial time algorithm to get a minimum fill-in for HHD-free graphs is due to [4].

7 Acknowledgements

I am very grateful for fruitful discussions with Ton Kloks.

References

[1] Arnborg, S., D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.* **8**, (1987), pp. 277–284.

[2] L. Babel, *Triangulating Graphs with Few P_4s*, submitted.

[3] H.J. Broersma, E. Dahlhaus, T. Kloks, *A Linear Time Algorithm for Minimum Fill In and Tree Width for Distance Hereditary Graphs* , submitted.

[4] H.J. Broersma, E. Dahlhaus, T. Kloks, *Algorithms for the Treewidth and Minimum Fill-in of HHD-Free Graphs*, WG 97, LLNCS 1335, pp. 109-117.

[5] P. Bunemann, *A Characterization of Rigid Circuit Graphs*, Discrete Mathematics 9 (1974), pp. 205-212.

[6] A. Cournier and M. Habib, *A new linear algorithm for modular decomposition*, in CAAP '94: 19th International Colloquium, Lecture Notes in Computer Science, Sophie Tison, ed., 1994, pp. 68-82.

[7] E. Dahlhaus, *Minimum Fill-in and Treewidth for Graphs Modularly Decomposable into Chordal Graphs*, WG98, LLNCS 1517, pp. 351-358.

[8] Elias Dahlhaus, Jens Gustedt, Ross McConnell, *Efficient and Practical Modular Decomposition*, Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (1997), pp. 26-35.

[9] M. Farber, *Characterizations of Strongly Chordal Graphs*, Discrete Mathematics 43 (1983), pp. 173-189.

[10] A. Frank, *Some Polynomial Time Algorithms for Certain Graphs and Hypergraphs*, Proceedings of the 5th British Combinatorial Conference, Congressus Numerantium XV, Untilitas Mathematicae, Winnipeg (1976), pp. 211-226.

[11] C.T. Hoáng and N. Khouzam, *On Brittle Graphs*, Journal of Graph Theory 12 (1988), pp. 391-404.

[12] R. M. McConnell and J. P. Spinrad, *Linear-time modular decomposition and efficient transitive orientation of undirected graphs*, in Proceedings of the fifth annual ACM-SIAM Symposium on Discrete Algorithms, D. D. Sleator et al., eds., Society of Industrial and Applied Mathematics (SIAM), 1994, pp. 536–545.

[13] Parra, A., Scheffler, P., *How to use minimal separators for its chordal triangulation*, Proceedings of the 20^{th} International Symposium on Automata, Languages and Programming (ICALP'95), Springer-Verlag Lecture Notes in Computer Science **944**, (1995), pp. 123–134.

[14] D. Rose, *Triangulated Graphs and the Elimination Process*, Journal of Mathematical Analysis and Applications 32 (1970), pp. 597-609.

[15] R. Tarjan, M. Yannakakis, *Simple Linear Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs*, SIAM Journal on Computing 13 (1984), pp. 566-579.
Addendum: SIAM Journal on Computing 14 (1985), pp. 254-255.

[16] M. Yannakakis, Computing the Minimum Fill-in is NP-complete, *SIAM Journal on Algebraic and Discrete Methods* 2 (1981), pp. 77-79.