Making an Arbitrary Filled Graph Minimal by Removing Fill Edges ⋆

Jean R. S. Blair[1] and Pinar Heggernes[2] and Jan Arne Telle[2]

[1] United States Military Academy, West Point, NY
[2] Department of Informatics, University of Bergen, Norway

Abstract. We consider the problem of removing fill edges from a filled graph G' to get a minimal chordal supergraph M of the original graph G; thus $G \subseteq M \subseteq G'$. We show that a greedy strategy can be applied if fill edges are processed for removal in the reverse order of their introduction. For a filled graph with f fill edges and e original edges, we give a simple $O(f(e+f))$ algorithm which solves the problem and computes a corresponding minimal elimination ordering. We believe that in practice the runtime of our algorithm is usually better than the worst-case bound of $O(f(e+f))$.

1 Introduction

For any graph G and an ordering α of its vertices, there is an associated set of edges called the *fill* that, when added to G, results in a chordal graph $(G; \alpha)$.[3] The goal of finding orderings of the vertices that produce a small fill has been studied by researchers in many areas of computer science (*e.g.,* data-base management systems [2, 14], knowledge-based systems [5, 7], computer vision [3] and the solution of sparse symmetric systems of linear equations [8, 9, 10, 11, 12]) and remains an important problem.

In a central 1976 paper Rose, Tarjan and Lueker [13] gave an algorithm which finds a *minimal* fill of a graph G in $O(ne)$ time, where n and e are respectively the number of vertices and edges in G. To date, this algorithm, which is called LEX-M, has the best known running time for finding a minimal chordal supergraph of an arbitrary graph. An efficient parallel algorithm for the same problem is given in [4]. Yannakakis showed in [15] that finding the *minimum* fill for an arbitrary graph is $\mathcal{NP}$-hard. Several heuristics have been proposed for finding elimination orderings producing small fill. The two most famous and practically useful methods are called *minimum degree* and *nested dissection* (see [6] for a survey). For graphs of bounded maximum degree, nested dissection based on near-optimal node separators gives fill which is within a ploylogarithmic factor of the minimum fill [1].

We consider the more restricted problem of, given a graph G and an arbitrary ordering α, finding a graph M that is both a minimal chordal supergraph of

⋆ This research was supported in part by the Norwegian Research Council and was conducted while the first author was visiting the University of Bergen, Norway.

[3] All terms are formally defined in Section 2.

G, and a subgraph of the filled graph $(G;\alpha)$. We also find a related minimal ordering β so that $(G;\beta) = M$. Minimal orderings are desirable since any perfect elimination ordering of the resulting filled graph, when applied to the original graph, produces the same filled graph. In other words, if β is a minimal ordering of a graph G then for any perfect elimination ordering γ of $(G;\beta)$ we have $(G;\gamma) = (G;\beta)$. The problem we consider is motivated by the fact that the minimal orderings produced by the LEX-M algorithm (and other breadth-first-search-based algorithms) typically result in a large amount of fill relative to that produced by practical low-fill heuristic algorithms such as minimum degree and nested dissection, and by the fact that these heuristic algorithms usually produce non-minimal fill. So to get a minimal low fill ordering of a graph G, first find a non-minimal low fill ordering α produced by a heuristic algorithm, and then run our algorithm which removes redundant fill until the remaining fill is minimal.

The main contributions of this paper are two-fold: First, our Theorem 7 shows that by considering the fill edges in the reverse order to that in which they were introduced, each edge need only be considered as a candidate for removal once. This property is significant since, in general, the removal of a fill edge can both add to and delete from the set of remaining "removable edges." Second, using this result, we develop an $O(f(e+f))$ time algorithm that, given both G and α, will produce an ordering β such that the filled graph $(G;\beta)$ is minimal and is a subgraph of $(G;\alpha)$. Here, f and e are respectively the number of fill edges in $(G;\alpha)$ and the number of edges in G. Furthermore, the time complexity of our algorithm depends on structural properties of the filled graph and is in many cases better than the worst-case bound of $O(f(e+f))$.

The paper is organized as follows. Section 2 formally defines terms used in the paper. Section 3 characterizes redundant fill edges. The new algorithm and its proof of correctness are found in Section 4 with an analysis of the time complexity. Section 5 contains concluding remarks and some questions raised by our results.

2 Definitions and Notation

We start with some standard graph terminology. We consider undirected, simple graphs. For a graph G, the vertex and edge sets are denoted by respectively $V(G)$ and $E(G)$. $N_G(v)$ is the set of neighbors of v in G. A *path* from v_1 to v_k is a sequence of vertices $v_1, v_2, ..., v_k$ that are connected by the edges $v_1v_2, v_2v_3, ..., v_{k-1}v_k$. We also use $v_1 \to v_k$ to denote a path from v_1 to v_k. A *cycle* is a path whose first and last vertices are the same. An edge in G is called a *chord* of a cycle if it joins two nonconsecutive vertices on the cycle. A graph is *chordal* if every cycle of length at least four has a chord. For a set S of vertices in G, the subgraph of G induced by S is denoted by $G[S]$. The graph $G \setminus S$ is the graph $G[V(G) \setminus S]$. For a set K of edges in G, the graph $G \setminus K$ is the result of removing the edges in K from G. A *supergraph* of G is a graph which contains G as a subgraph.

For a graph G with $|V(G)| = n$, an *elimination ordering* of G is a bijection

$\alpha : V(G) \leftrightarrow \{1, 2, ..., n\}$. For ease of presentation, we will also refer to α as a sequence, $\alpha = v_1, v_2, ..., v_n = \alpha^{-1}(1), \alpha^{-1}(2), ..., \alpha^{-1}(n)$. Thus v_i denotes the vertex v such that $\alpha(v) = i$.

Associated with α is a sequence of supergraphs of G, defined as follows. $G_0 = G$ and for $1 \leq i \leq n$, G_i is the graph obtained by adding edges to G_{i-1} so that all vertices in $N_{G_{i-1}}(v_i) \cap \{v_{i+1}, ...v_n\}$ are pairwise adjacent. This step is called the *elimination* of vertex v_i, and the whole process of obtaining G_n from G is called the *elimination process*. Note that vertices are not removed from the graph: $V(G) = V(G_i)$, $1 \leq i \leq n$. The new edges added are called *fill edges*, and $F_i = E(G_i) \setminus E(G_{i-1})$ is the set of fill edges created by the elimination of v_i. We use C_i to denote the resulting clique induced by v_i and its higher numbered neighbors. The graph G_n is the *filled graph* of G for elimination ordering α. We also use $(G; \alpha)$ to denote the filled graph G_n. All filled graphs are chordal.

An elimination ordering α on G is *minimal* if the resulting $(G; \alpha)$ is a *minimal chordal* supergraph of G, *i.e.,* no strict subgraph of $(G; \alpha)$ containing G is chordal, equivalently for no ordering γ is $(G; \gamma)$ a strict subgraph of $(G; \alpha)$. Then $(G; \alpha)$ is also referred to as a *minimal filled graph* of G.

A vertex is *simplicial* in a graph G if its neighbors induce a clique. Note that v_i is simplicial in $G_{i-1}[\{v_i, ..., v_n\}]$ if and only if $F_i = \emptyset$. An elimination ordering on G is *perfect* if no fill edges are created, that is, if $F_i = \emptyset$ for $1 \leq i \leq n$. Chordal graphs are exactly the class of graphs that have perfect elimination orderings.

3 Greedy Removal of Redundant Fill Edges

In this section we develop some properties of fill edges related to the order of their introduction, with the goal of finding an algorithm to remove redundant fill edges. The following result (Lemma 4 of [13]) gives another characterization of fill edges.

Lemma 1. *[13] Let G be a graph and α an elimination ordering of its vertices. Then uv is an edge of $(G; \alpha)$ if and only if $uv \in E(G)$ or there exists a path $u = u_0, u_1, ..., u_{k+1} = v$ in G such that for each $1 \leq i \leq k$ we have $\alpha(u_i) < \min\{\alpha(u), \alpha(v)\}$.*

The same paper also gives (Theorem 2 of [13]) an alternative characterization of minimal elimination orderings.

Theorem 2. *[13] Let G be a graph and α an elimination ordering of its vertices. Then α is a minimal elimination ordering if and only if each fill edge is the unique chord of a 4-cycle in $(G; \alpha)$.*

We will need the following easy corollary.

Corollary 3. *Let G be a graph and $\alpha = v_1, ..., v_n$ be an elimination ordering of its vertices. If a fill edge $uv \in F_i$, created by elimination of v_i, is the unique chord of a 4-cycle in $(G; \alpha)$ then uv is the unique chord of a 4-cycle u, v_i, v, x, u in $(G; \alpha)$ with $\alpha(x) > i, \alpha(u) > i$ and $\alpha(v) > i$.*

Proof. Let u, y, v, x, u be a 4-cycle of $(G; \alpha)$ in which uv is the only chord. Since uv is created by elimination of v_i we must have $\alpha(u) > i$, $\alpha(v) > i$, $\alpha(y) \geq i$ and $\alpha(x) \geq i$. If either $y = v_i$ or $x = v_i$ we are done. If this is not the case, note that either yv_i or xv_i is not an edge of $(G; \alpha)$ since otherwise we would have xy a fill edge. As u and v are both neighbors of v_i, we can find a 4-cycle as stated in the corollary. □

Observe that Corollary 3 and Theorem 2 can be used to easily detect if a given elimination ordering is minimal.

Definition 4. Let G' be a chordal supergraph of a graph G. A *candidate* edge is an edge $uv \in E(G') \setminus E(G)$ which is not the unique chord of any 4-cycle in G'.

Since removing a single edge from a chordal graph results in either a chordal graph or a graph with a chordless 4-cycle, it is clear that removing any candidate edge cannot destroy chordality. The following result is implicit in [13].

Proposition 5. *[13] A non-minimal chordal supergraph G' of a graph G has at least one candidate edge. Any candidate edge can be removed without destroying chordality.*

This result immediately leads to an algorithm for finding a minimal filled graph by removing fill edges from a filled graph: repeatedly remove candidate edges until no candidate edges remain. Unfortunately, the set of candidate edges changes as edges are removed, with new candidate edges being introduced and old candidate edges ceasing to be candidates. For a very simple example, consider the path on 4 vertices P_4 with edges ab, bc, cd and elimination order b, c, d, a creating the fill $F_1 = \{ac\}$ and $F_2 = \{ad\}$. The fill edge ac is not a candidate for removal but ad is. However, after removing ad the fill edge ac becomes a candidate for removal.

This non-monotonicity property of the set of candidate edges seems to be the main obstacle in obtaining an efficient implementation of this algorithm. As we show below, we can partially avoid this by processing fill edges in the reverse fill order of that in which they were introduced. In particular, the following strategy will produce a minimal chordal supergraph:

for $i = n$ **downto** 1 **do**
  while there is a candidate edge in F_i **do** remove it from the filled graph;

We first give a lemma and then a theorem that will provide the basis for the correctness of this strategy.

Lemma 6. *Let $\alpha = v_1, ..., v_n$ be an elimination order of a graph G. Let M_i be both a subgraph of G_n and a minimal chordal supergraph of G_{i-1}. Then M_i has a perfect elimination ordering $\gamma = w_1, w_2, ..., w_n$ with $(G; \gamma) = M_i$ and and $v_k = w_k$ for all $k = 1, ..., i-1$.*

Proof. Since both G_{i-1} and G_n are associated with $\alpha = v_1, ..., v_n$, we have for all $k = 1, ..., i-1$ the vertex v_k simplicial in both the graphs $G_{i-1}[\{v_k, v_{k+1}, ..., v_n\}]$ and $G_n[\{v_k, v_{k+1}, ..., v_n\}]$. Note also that v_k has the same neighbors in both these graphs.

Hence v_k is also simplicial in M_i since M_i is a supergraph of G_{i-1} and a subgraph of G_n. For chordal graphs it is well-known that the elimination order resulting from repeatedly eliminating a vertex which is simplicial in the graph induced by non-eliminated vertices, is a perfect elimination ordering. We can therefore find a perfect elimination ordering γ of M_i with $\gamma = w_1, ..., w_n$ and $v_k = w_k$ for $k = 1, ..., i-1$.

We now show that $(G;\gamma) = M_i$. Eliminating $v_1, ..., v_{i-1}$ in this order in G or G_{i-1} will in either case give us G_{i-1}. Since M_i is a minimal chordal supergraph of G_{i-1} we must have $(G_{i-1};\gamma) = M_i$. Hence, $(G;\gamma) = M_i$ since the initial sequence of γ is $v_1, ..., v_{i-1}$. □

We can now state the main theorem of this section.

Theorem 7. *Let $\alpha = v_1, ..., v_n$ be an elimination order of a graph G. Let M_i be both a subgraph of $(G;\alpha)$ and a minimal chordal supergraph of G_{i-1}. For any graph M which is both a subgraph of M_i and a chordal supergraph of G, we have $E(M_i) \setminus E(G_{i-1}) \subseteq E(M)$.*

Proof. Observe that $G \subseteq G_{i-1} \subseteq M_i \subseteq (G;\alpha)$ and $G \subseteq M \subseteq M_i$. By Lemma 6, we know that M_i has a perfect elimination ordering $\gamma = w_1, w_2, ..., w_n$ with $v_k = w_k$ for all $k = 1, ..., i-1$ and we also know that $(G_{i-1};\gamma) = M_i$. The statement in the theorem says that if the edge uv of M_i is a fill edge created, in the process giving $(G_{i-1};\gamma) = M_i$ from G_{i-1}, by the elimination of some w_j with $i \leq j \leq n$, then uv must be an edge of M.

We prove this by contradiction. Assume that uv is the latest introduced fill edge violating the condition. In other words, uv is created in the process giving $(G_{i-1};\gamma) = M_i$ by $w = w_j$, uv is not an edge of M, and every other fill edge of this elimination process created by any w_k with $k > j$ is an edge of M. Since M_i is a minimal chordal supergraph of G_{i-1}, and uv is a fill edge created by w, by Theorem 2 and Corollary 3 there must exist a vertex x which in M_i is not adjacent to w but is adjacent to both u and v, with $\gamma(u) > \gamma(w)$, $\gamma(v) > \gamma(w)$ and $\gamma(x) > \gamma(w)$.

We will show that there exists, in $M \subseteq M_i$, a path from w to x whose internal vertices have γ-order less than w and x. We will thereby arrive at a contradiction since, with γ being a perfect elimination ordering of M_i, such a path would imply, by Lemma 1, that wx is an edge of M_i. But w and x are not adjacent in M_i.

Since neither uv nor wx are edges in the chordal graph M, at least one of wu, ux, xv and vw is not an edge of M. But M is a supergraph of G so any of wu, ux, xv and vw which is not an edge of M must be a fill edge in the elimination process $(G;\gamma)$ giving M_i. By assumption, uv is the latest fill edge in this elimination process which is not in M, so any of wu, ux, xv and vw which is not an edge of M, must be a fill edge of M_i created earlier than uv. (Note it could not be created at the same time as uv.) Therefore, by Lemma 1 there must exist paths $w \to u, u \to x, x \to v$, and $v \to w$ in $G \subseteq M$ whose internal vertices (if any) are earlier in the γ-order than w and thus also earlier than u, x, v. Consider the combined paths $u \to w \to v$ and $u \to x \to v$. If these two paths intersect in an internal vertex y then we find a path $w \to y \to x$ giving the desired contradiction. Otherwise, consider the shortest path $u \to w' \to v$ in M from u to v using only vertices on the $u \to w \to v$ path and the shortest path $u \to x' \to v$ in M from u to v using only vertices on the $u \to x \to v$ path. These paths must have an internal vertex since uv is not an edge of M, and the vertices on these paths therefore induce a subgraph of M containing a k-cycle with $k \geq 4$. Since M is chordal, there is a chord on this cycle and this chord must connect an internal vertex y' in $u \to x' \to v$ with an internal vertex y'' in $u \to w' \to v$. But then we find a path $w \to y' \to y'' \to x$ giving the desired contradiction. (Note that this argument allows for $w = y'$ or $x = y''$.) $\Box$

4 The Algorithm

In this section, we develop an algorithm which given a graph G and an elimination ordering α of G, finds a graph M which is both a minimal chordal supergraph of G and also a subgraph of $(G; \alpha)$.

After computing $G_n \equiv (G; \alpha)$, and finding C_i and F_i for $i = 1, ..., n$ from G_n in a straightforward manner, our algorithm proceeds as follows: The algorithm has n iterations. Initially, we set $M = G_n$. Starting from $i = n$ and going backwards, at each iteration i, redundant fill edges in F_i introduced by the elimination of v_i are removed. By Theorem 7, we know that the remaining edges of F_i need not be considered for removal at later iterations. The algorithm has n iterations for simplicity, but actually $n - 3$ would suffice since F_n and F_{n-1} are empty and F_{n-2} can always be removed.

The full algorithm is given in Figure 1. The subroutine that checks an edge $uv \in F_i$ for its candidacy for removal is called CandidateEdge, and is based on Corollary 3. The subroutine LEX-M is used to decide which of the candidate edges in F_i can be removed and which must stay to preserve chordality. W_i is the subgraph of C_i on which we run LEX-M to find out which edges in $\mathit{Candidate}(i)$ are necessary. The original LEX-M was introduced by Rose, Tarjan and Lueker in [13], and finds an elimination ordering resulting in a minimal chordal supergraph of a given graph. Our version of LEX-M(W_i) returns the set $\mathit{KeepFill}(i) \subseteq \mathit{Candidate}(i)$ of fill edges whose addition to W_i produces a minimal chordal supergraph of W_i. Note that $W_i \cup \mathit{Candidate}(i)$, if nonempty, is a clique and a subgraph of C_i.

Since the resulting graph M is a minimal chordal supergraph of G, every perfect elimination ordering β of M gives $(G; \beta) = M$. A linear-time algorithm for finding a perfect elimination ordering of a chordal graph is given in [13] and is called LEX-P. Figure 2 illustrates how the algorithm processes a graph on 7 edges, with details of the calls of LEX-M in Figure 3. We first prove correctness of the algorithm and then consider its time complexity.

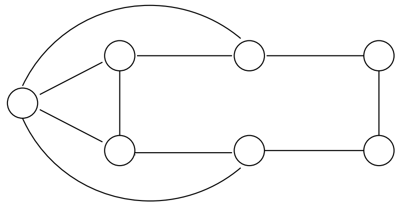
Algorithm MinimalChordal (G, α);
Input: A graph G and an elimination ordering $\alpha = v_1, ..., v_n$ of G.
Output: 1. A chordal graph M which is both a minimal chordal supergraph of G and a subgraph of the filled graph $(G; \alpha)$.
2. A minimal elimination order β of G s.t. M is the filled graph $(G; \beta)$.

```
begin
    Find (G;α) and C_i, F_i for i = 1, 2, ..., n;
    M = (G;α);
    for i = n downto 1 do
        Candidate(i) = ∅;
        Incident(i) = ∅;
        for all edges uv ∈ F_i do
            if CandidateEdge(uv, i, M) then
                Candidate(i) = Candidate(i) ∪ {uv};
                Incident(i) = Incident(i) ∪ {u, v};
            end-if;
        if Candidate(i) ≠ ∅ then
            W_i = C_i[Incident(i)] \ Candidate(i);
            KeepFill(i) = LEX-M(W_i);
            M = M \ (Candidate(i) \ KeepFill(i));
        end-if;
    end-for;
    return M and β = LEX-P(M);
end;
```

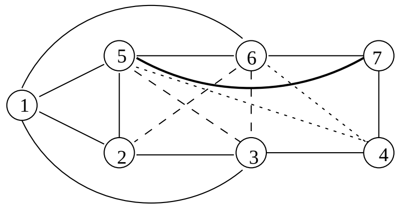
Function CandidateEdge(uv, i, M): **boolean**;
Input: An edge $uv \in F_i$ and the graph M.
Output: Returns **true** if uv is a candidate to be removed from M, **false** o.w.

```
begin
    cand = true;
    for each neighbor x of u do
        if α(x) > i and xv ∈ E(M) and xv_i ∉ E(M) then
            cand = false;
    return cand;
end;
```

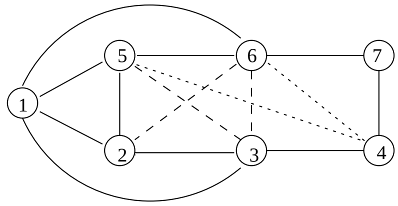
Fig. 1. Algorithm MinimalChordal and Function CandidateEdge.



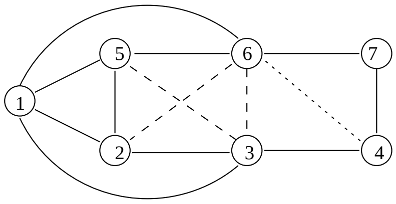
1. The original graph G.



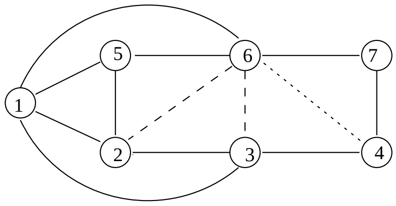
2. The filled graph of G resulting from the shown elimination ordering. The fill edges introduced by vertices 1, 3 and 4 are drawn with respectively dashed, dotted and thick lines.



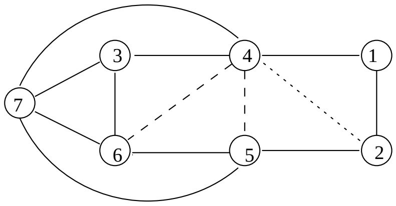
3. After removal of fill introduced by vertex 4. The edge (5,7) was a candidate edge and was removed by LEX-M.



4. After removal of fill introduced by vertex 3. The edge (4,5) was a candidate edge and was removed by LEX-M, whereas (4,6) was not a candidate edge since (6,7) and (4,7) are edges, and (3,7) is not an edge.



5. After removal of fill introduced by vertex 1. Edges (2,6) and (3,5) were candidate edges, whereas (3,6) was not a candidate edge since (6,4) and (3,4) are edges and (2,4) is not. One of the candidate edges can be removed by LEX-M, and (3,5) was chosen.



6. A perfect elimination ordering on the resulting minimal chordal supergraph of G.

Fig. 2. An example illustrating the algorithm. The calls of LEX-M in Steps 4 and 5 are shown in detail in Figure 3.

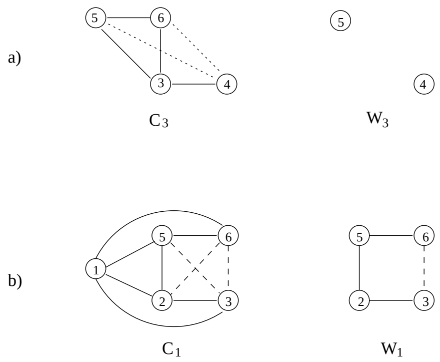


Fig. 3. Steps 4 and 5 of the previous example (Figure 2) shown in detail. a) $Candidate(3) = \{(4,5)\}$, and $Incident(3) = \{4,5\}$. The subgraph W_3 is defined as $C_3[Incident(3)] \setminus Candidate(3)$. LEX-M($W_3$) returns no fill, hence $(4,5)$ is removed from M. b) $Candidate(1) = \{(2,6),(3,5)\}$, and $Incident(1) = \{2,3,5,6\}$. LEX-M($W_1$) returns $(2,6)$ as fill, thus only $(3,5)$ can be removed from M as redundant fill.

Theorem 8. *Algorithm MinimalChordal on input G and $\alpha = v_1, ..., v_n$ finds a graph M which is both a minimal chordal supergraph of G and a subgraph of $(G;\alpha)$.*

Proof. Denote the graph M at the beginning of iteration i of the main loop of the algorithm by M_{i+1}, and at the end of iteration i by M_i. Since $M_i \subseteq M_{i+1}$ and M_{n+1} is initialized to $G_n \equiv (G;\alpha)$ it is clear that each M_i is a subgraph of $(G;\alpha)$. In addition, the algorithm has the loop invariant: "The graph M_i is a minimal chordal supergraph of G_{i-1}." We show this by reverse induction on i from $n+1$ to 1. The loop invariant is clearly true initially for the graph $M_{n+1} = G_n$. The edges $E(M_{i+1})$ are of four types:

(1) edges belonging to the original graph G.

(2) fill edges that were introduced before the elimination of v_i which will be considered for removal at later iterations.

(3) fill edges that were introduced after the elimination of v_i which have not been removed at earlier iterations.

(4) fill edges belonging to F_i.

The graphs G_i and G_{i-1} both contain all the edges of types 1 and 2, so these edges must belong to any chordal supergraph of G_{i-1}. Since M_{i+1} is a subgraph of G_n and a minimal chordal supergraph of G_i we know by Theorem 7 that no edges of Type 3 can be removed from M_{i+1} and still give a chordal supergraph of $G \subseteq G_{i-1}$.

Hence, we need only show that (A) M_i is chordal and that (B) any edge uv of F_i which remains in M_i is not a candidate for removal. Consider Case (B) first. By Theorem 2 it suffices to show that any such edge uv is the only chord of a 4-cycle in M_i. There are two cases: either (B.1) $uv \in Candidate(i)$ or (B.2) not.

(B.1) At the beginning of iteration i, the neighbors among $\{v_{i+1}, ..., v_n\}$ in M_{i+1} of vertex v_i induce a clique C_i, and the fill edges F_i are all contained in this clique. The edges in F_i that are candidates for removal are identified as $Candidate(i)$, and the vertices which are incident to at least one candidate edge, are stored in $Incident(i)$. We set W_i to be the subgraph of C_i induced by the vertices in $Incident(i)$ minus the edges in $Candidate(i)$. Candidate edges of F_i may be removed in the call of LEX-M(W_i), which gives us a set $KeepFill(i) \subseteq Candidate(i)$ of fill edges that when added to W_i gives a minimal chordal supergraph of W_i. The edges in $Candidate(i) \setminus KeepFill(i)$ are then removed from M_{i+1} to give us M_i. An edge $uv \in KeepFill(i)$ is guaranteed, by the correctness of algorithm LEX-M and Theorem 2, to be a unique chord of a 4-cycle in the graph $W_i \cup KeepFill(i)$ and therefore also in M_i, since $W_i \cup KeepFill(i)$ is an induced subgraph of M_i.

(B.2) If uv is found not to be a candidate in the call CandidateEdge(uv, i, M_{i+1}) then it is because there exists a vertex x which in M_{i+1} is a neighbor of both u and v but is not a neighbor of v_i. Note that the 4-cycle v_i, u, x, v, v_i contains only edges of Type 1, 2 or 3 and uv is therefore a unique chord of this 4-cycle also in M_i.

(A) It remains to show that M_i is chordal, which we do by contradiction. Let S be the vertices on a shortest chordless cycle of length at least 4 of M_i. We know that there must exist at least two vertices a, c in S such that $ac \in Candidate(i) \setminus KeepFill(i)$, since $M_{i+1} = M_i \cup (Candidate(i) \setminus KeepFill(i))$ is chordal. Also S must contain at least one vertex $s \notin V(C_i)$ since the graph $M_i[V(C_i)]$ is chordal ($M_i[V(C_i)]$ contains the minimal chordal supergraph of W_i computed by LEX-M and the vertices in $V(C_i) \setminus Incident(i)$ which are incident to every other vertex in $V(C_i)$.) Moreover, every vertex $x \in S$ must have $\alpha(x) \geq i$ since M_i contains all fill edges $F_1, ..., F_{i-1}$, and the cycle induced by S contains no chords. We can therefore find a path $a, b_1, ..., b_k, c$ in $M_i[S]$ with a and c in $V(C_i)$ and $ac \in Candidate(i) \setminus KeepFill(i)$ and $b_j, 1 \leq j \leq k$ not in $V(C_i)$ and $k \geq 1$. But then we have $a, b_1, ..., b_k, c, v_i, a$ inducing a chordless $k+3$-cycle in M_{i+1}, since none of b_j is in $V(C_i)$, contradicting the chordality of the graph M_{i+1}.

The loop invariant therefore holds as claimed. Note that this establishes correctness of the algorithm as upon termination the returned graph M_1 is both a minimal chordal supergraph of $G_0 = G$ and a subgraph of $G_n \equiv (G; \alpha)$. □

Theorem 9. *Let $\alpha = v_1, v_2, ..., v_n$ be an elimination ordering of a connected graph G, with $n = |V(G)|$, $e = |E(G)|$ and $f = \sum_{i=1}^{n} |F_i|$. The time complexity of Algorithm MinimalChordal(G, α) is $O(f(e+f))$.*

Proof. Computing G_n (see [14]) and the call of LEX-P (see [13]) both take time linear in the size of the filled graph. From G_n, the sets F_i and C_i can be computed in time $O(nf)$ by simply examining every fill edge for every vertex. The algorithm has n iterations. At each iteration i, CandidateEdge is called $|F_i|$ times, and LEX-M is called once. The time complexity of CandidateEdge(uv, i, M) is $O(|N_G(u)|) = O(n)$. The time complexity of LEX-M(W_i) is $O(|V(W_i)||E(W_i)|)$, (see [13]). Thus for the whole algorithm, we get:

$$O\left(\sum_{i=1}^{n}(|V(W_i)||E(W_i)| + |F_i|n)\right)$$

By the definition of W_i, $|V(W_i)| = |Incident(i)| \leq 2|F_i|$. Clearly, $|E(W_i)| \leq (e+f)$. Thus, the time complexity of LEX-M(W_i) becomes $O(|F_i|(e+f))$. Since all the F_i are disjoint, $\sum_{i=1}^{n} |F_i|n = O(fn)$, and $\sum_{i=1}^{n} |F_i|(e+f) = O(f(e+f))$. Since G is connected, $n \leq (e+f)$, and the overall time complexity is $O(f(e+f))$ $\square$

We would like to emphasize that the time complexity $O(f(e+f))$ is usually an overestimate and the sum in the proof of Theorem 9 is often less than this bound. A brief discussion about this is included in the next concluding section.

5 Conclusion

As mentioned in the Introduction, minimal low fill orderings are desirable, but minimal fill is not necessarily low fill. Practical algorithms like minimum degree and nested dissection usually produce low fill orderings, but not minimal fill orderings. Therefore we assumed a filled graph (with low fill) already given and considered the problem of removing fill edges to produce a minimal filled graph.

We have given a strategy to remove candidate fill edges so that each fill edge is checked for candidacy exactly once, avoiding backtracking. The time complexity of our algorithm is dependent on the number of fill edges we start with. In some cases this number is already low, in other cases it is high and our algorithm may remove many redundant fill edges. The upper bound given on the time complexity is met only if the subgraphs $W_1, W_2, ..., W_n$ overlap heavily so that the calls of LEX-M involve some large part of the graph several times. However, from examples we have studied it seems that these subgraphs usually do not overlap very much. In the example of Figure 3 we see that the subgraphs W_i do not overlap on edges at all.

From the results given in this paper the following questions arise:

1. Can the time complexity of our algorithm be improved if we assume that the given elimination ordering α was produced by the minimum degree algorithm? What if α was produced by the nested dissection algorithm?
2. Can the time complexity of our algorithm be expressed as a function of the number of fill edges removed?
3. Does our algorithm have an implementation which is fast enough to be useful in practice?
4. Does Theorem 7 have other interesting implications?

Acknowledgement

The authors are indebted to Barry Peyton for suggesting the problem of finding a minimal chordal supergraph from a non-minimal chordal supergraph.

References

1. A.Agrawal, P.Klein, and R.Ravi. Cutting down on fill using nested dissection: provably good elimination orderings. In J. A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Sparse matrix computations: Graph theory issues and algorithms*, pages 31–55. Springer Verlag, 1993. IMA Volumes in Mathematics and its Applications, Vol. 56.
2. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database systems. *J. Assoc. Comput. Mach.*, 30:479–513, 1983.
3. F. R. K. Chung and D. Mumford. Chordal completions of planar graphs. *J. Comb. Theory*, 31:96–106, 1994.
4. E. Dahlhaus and M. Karpinski. An efficient parallel algorithm for the minimal elimination ordering of an arbitrary graph. *Proceedings FOCS*, pages 454–459, 1989.
5. R. E. England, J. R. S. Blair, and M. G. Thomason. Independent computations in a probablistic knowledge-based system. Technical Report CS-90-128, Department of Computer Science, The University of Tennessee, Knoxville, Tennessee, 1991.
6. A. George and J.W-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
7. S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *J. Royal Statist. Soc., ser B*, 50:157–224, 1988.
8. J. G. Lewis, B. W. Peyton, and A. Pothen. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput.*, 10:1156–1173, 1989.
9. J. W-H. Liu and A. Mirzaian. A linear reordering algorithm for parallel pivoting of chordal graphs. *SIAM J. Disc. Math.*, 2:100–107, 1989.
10. T. Ohtsuki, L.K.Cheung, and T.Fujisawa. Minimal triangulation of a graph and optimal pivoting ordering in a sparse matrix. *J. Math. Anal. Appl.*, 54:622–633, 1976.
11. B. W. Peyton. *Some applications of clique trees to the solution of sparse linear systems*. PhD thesis, Dept. of Mathematical Sciences, Clemson University, 1986.
12. D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, 1972.
13. D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266–283, 1976.
14. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.
15. M. Yannakakis. Computing the minimum fill-in is np-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.