# Maximum Cardinality Search for Computing Minimal Triangulations of Graphs*

Anne Berry† Jean R. S. Blair‡ Pinar Heggernes§

Barry W. Peyton¶

**Abstract**

We present a new algorithm, called MCS-M, for computing minimal triangulations of graphs. Lex-BFS, a seminal algorithm for recognizing chordal graphs, was the genesis for two other classical algorithms: LEX M and MCS. LEX M extends the fundamental concept used in Lex-BFS, resulting in an algorithm that not only recognizes chordality, but also computes a minimal triangulation of an arbitrary graph. MCS simplifies the fundamental concept used in Lex-BFS, resulting in a simpler algorithm for recognizing chordal graphs. The new algorithm MCS-M combines the extension of LEX M with the simplification of MCS, achieving all the results of LEX M in the same time complexity.

## 1 Introduction

An important and widely studied problem in graph theory with applications in sparse matrix computations [10, 14, 16, 17, 18], database management [1, 19], knowledge based systems [11], and computer vision [7], is that of adding as few edges as possible to a given graph so that the resulting *filled* graph is chordal. Such a filled graph is called a *minimum triangulation* of the input graph. Computing a minimum triangulation is NP-hard [20]. In this paper, we study the polynomially computable alternative of finding a minimal triangulation. A *minimal triangulation* $H$ of a given graph $G$ is a triangulation such that no proper subgraph of $H$ is a triangulation of $G$.

*A preliminary version of this work appeared as [3]. This collaboration was initiated while the first two authors were visiting the University of Bergen.

†LIMOS, UMR CNRS 6158, Université Clermont-Ferrand II, F-63177 Aubiere, France, berry@isima.fr

‡Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, New York, USA, Jean.Blair@usma.edu

§Department of Informatics, University of Bergen, N-5020 Bergen, Norway, Pinar.Heggernes@ii.uib.no

¶Natural Sciences and Mathematics Division, Dalton State College, Dalton, Georgia, USA, bpeyton@em.daltonstate.edu

Several practical algorithms exist for finding minimal triangulations [2, 5, 8, 12, 15, 18]. One such classical algorithm, called LEX M [18], is derived from the Lex-BFS (lexicographic breadth-first search) algorithm [18] for recognizing chordal graphs.[1] Both Lex-BFS and LEX M use lexicographic labels of the unprocessed vertices. As processing continues, the remaining labels grow, each potentially reaching a length proportional to the number of vertices in the graph. Lex-BFS adds to the labels of the neighbors of the vertex being processed, while LEX M adds to the labels of both neighbors and other vertices that can be reached along special kinds of paths from the vertex being processed. These special kinds of paths are precisely the paths that produce fill, and the labels in LEX M are exactly the higher numbered neighbors in the current filled graph. Interestingly, the simple extension of adding to labels based on reachability along such fill paths, rather than only along single edges, results in an algorithm that produces minimal triangulations.

The adjacency labeling concepts developed for Lex-BFS have proved to be central in the understanding of chordal graphs and triangulations. Tarjan and Yannakakis later came up with the surprising result that for the case of recognizing chordality, knowing the specific processed neighbors (i.e., labels) is not necessary; one need only maintain and compare the number of processed neighbors [19]. This was an important achievement resulting in a significantly simpler implementation of Lex-BFS, known as the MCS (maximum cardinality search) algorithm. A natural question that arises is whether or not cardinality comparisons are also sufficient for the case of minimal triangulations. That is, is there a significantly simplified implementation of LEX M that uses only the cardinality of processed vertices that can be reached along special kinds of paths that turn out to be fill paths? Or, equivalently, can MCS be extended from neighbors to paths in order to yield a minimal triangulation algorithm, in analogy with the extension from Lex-BFS to LEX M? In this paper, we introduce an algorithm called MCS-M to fill exactly this gap.

The relationships between the four algorithms discussed thus far are summarized in Figure 1. In the figure, the algorithms on the left recognize chordal graphs while those on the right produce minimal triangulations of arbitrary graphs, as well as recognize chordality. Both algorithms on the left have time complexity $O(n+m)$; both algorithms on the right have time complexity $O(nm)$.

This paper is organized as follows. In the next section we give the necessary background in graph terminology, assuming that the reader is familiar with standard graph notation. Included in Section 2 is a classical characterization of minimal triangulations that forms the basis for our proofs of

[1] Lex-BFS was originally called LEX P by its authors in [18] and then later was called RTL in [19].

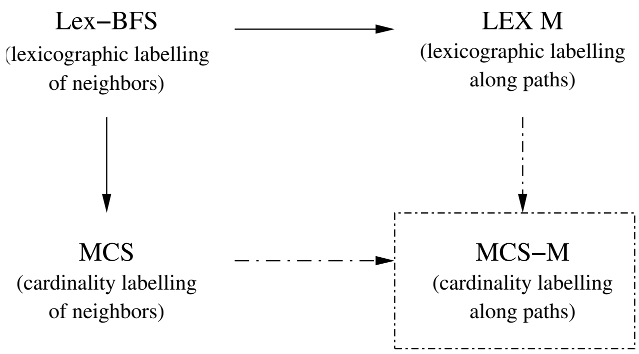


Figure 1: Relationships between algorithms. Solid arrows represent previous evolution. Dashed arrows represent the natural evolution to a new MCS-M algorithm.

correctness. The three algorithms that lead to the results in this paper are presented in Section 3. Section 4 describes the new minimal triangulation algorithm MCS-M, and proves its correctness. The proofs add insight into MCS and minimal triangulations in general, and lead to an interesting invariant of MCS, which is discussed in Section 5. We conclude with a time complexity discussion and several open questions in Section 6.

## 2 Background

All graphs in this work are undirected and finite. A graph is denoted by $G = (V, E)$, with $n = |V|$, and $m = |E|$. For a set $A \subseteq V$, $G(A)$ denotes the subgraph of $G$ induced by the vertices belonging to $A$. The *neighborhood* of a vertex $v$ in $G$ is $N_G(v) = \{u \mid uv \in E\}$, and the *closed neighborhood* of $v$ is $N_G[v] = N_G(v) \cup \{v\}$. Similarly, for a set $A \subseteq V$ of vertices in $G$, $N_G(A) = \cup_{v \in A} N_G(v) - A$, and $N_G[A] = N_G(A) \cup A$. When the graph $G$ is clear from the context, we will omit the subscript $G$.

A *clique* is a set of pairwise adjacent vertices. A vertex $v$ is *simplicial* if $N(v)$ is a clique. A *chord* of a cycle is an edge connecting two non-consecutive vertices of the cycle. A graph is *chordal*, or equivalently *triangulated*, if it contains no chordless cycle of length $\geq 4$. A *triangulation* of a graph $G = (V, E)$ is a chordal graph $G^+ = (V, E \cup F)$ that results from the addition of a set $F$ of *fill edges*, and $G^+$ is a *minimal triangulation* if $(V, E \cup F')$ fails to be chordal for every proper subset $F'$ of $F$.

The algorithm shown in Figure 2, called the *elimination game*, was first introduced by Parter [14]. Its input is a graph $G$ and an ordering $\alpha$ of $G$, where $\alpha(v) = i$ if $v$ is the $i^{th}$ vertex in ordering $\alpha$. The resulting filled graph $G^+_\alpha$ is a triangulation of $G$ [9]. The ordering $\alpha$ is a *perfect elimination ordering* if no fill edges are added during the elimination game, i.e. $G^+_\alpha = G$. Note that this is equivalent to choosing a simplicial vertex at each step of the

**Algorithm** Elimination Game
**Input:** A general graph $G = (V, E)$, and an ordering $\alpha$ of the vertices in $G$.
**Output:** The filled graph $G^+_\alpha$, which is a triangulation of $G$.
**begin**
  $G^0 = G$;
  **for** $i = 1$ **to** $n$ **do**
    Let $v$ be the vertex for which $\alpha(v) = i$;
    Let $F^i$ be the set of edges necessary to make $N_{G^{i-1}}(v)$ a clique in $G^{i-1}$;
    Obtain $G^i$ by adding the edges in $F^i$ to $G^{i-1}$ and removing $v$;
  $G^+_\alpha = (V, E \cup^n_{i=1} F^i)$;
**end**

Figure 2: The elimination game.

elimination game. Fulkerson and Gross [9] showed that the class of chordal graphs is exactly the class of graphs having perfect elimination orderings. Thus when the input graph $G$ is not chordal, no perfect elimination of it exists. However, if $G^+_\alpha$ is a minimal triangulation of $G$, then $\alpha$ is called a *minimal elimination ordering*, and if $G^+_\alpha$ is a minimum triangulation of $G$, then $\alpha$ is called a *minimum elimination ordering* on $G$.

The following theorem characterizes the edges of the filled graph.

**Theorem 2.1** (Rose, Tarjan, and Lueker [18]) *Given a graph $G = (V, E)$ and an elimination ordering $\alpha$ of $G$, $uv$ is an edge in $G^+_\alpha$ if and only if $uv \in E$ or there exists a path $u, x_1, x_2, ..., x_k, v$ in $G$ where $\alpha(x_i) < \min\{\alpha(u), \alpha(v)\}$, for $1 \leq i \leq k$.*

The authors of [18] also give several interesting characterizations of minimal triangulations. The following theorem, which characterizes minimal triangulations by their fill edges, will be used to prove the correctness of our algorithm.

**Theorem 2.2** (Rose, Tarjan, and Lueker [18]) *Given a graph $G = (V, E)$, a triangulation $H = (V, E \cup F)$ of $G$ is minimal if and only if every edge in $F$ is the unique chord of a 4-cycle in $H$.*

# 3 The Lex-BFS, LEX M, and MCS algorithms

The MCS algorithm (Figure 3) is a simple linear time algorithm that first processes an arbitrary vertex $v$ assigning $\alpha(v) = n$, and then continues generating an elimination ordering in reverse. MCS maintains, for each vertex $v$, an integer weight $w(v)$ that is the cardinality of the already processed neighbors of $v$. When given a chordal graph as input, MCS produces a perfect elimination ordering.

Lex-BFS has the same description as MCS, but uses labels that are lists of the already processed neighbors, instead of using weights. In the

**Algorithm** MCS
**Input:** A graph $G$.
**Output:** An ordering $\alpha$ of $G$.
**begin**
  **for** all vertices $v$ in $G$ **do**
    $w(v) = 0$;
  **for** $i = n$ **downto** 1 **do**
    Choose an unnumbered vertex $v$ of maximum weight $w(v)$;
    **for** all unnumbered vertices $u \in N(v)$ **do**
      $w(u) = w(u) + 1$;
    $\alpha(v) = i$;
**end**

Figure 3: Maximum Cardinality Search.

beginning $l(v) = \emptyset$ for all vertices. At step $n - i + 1$, an unnumbered vertex $v$ of lexicographically highest label is chosen to receive number $i$, and $i$ is added to the end of the label lists of all unnumbered neighbors of $v$.

LEX M is an extension of Lex-BFS that uses Theorem 2.1 to compute a minimal triangulation in the following way. When $v$ receives number $i$ at step $n - i + 1$, LEX M adds $i$ to the end of the label lists of all unnumbered vertices $u$ for which there exists a path between $v$ and $u$ consisting only of unnumbered vertices with lexicographically lower labels than those of $v$ and $u$. We will call such a path a *fill path*. Thus vertex $v$ appends its number $\alpha(v)$ to the label of every vertex $u$ which is connected to $v$ through a fill path, and $vu$ is an edge of the resulting minimal triangulation. Furthermore, the ordering generated by LEX M is a minimal elimination ordering of the input graph.

In the next section we show that using weights rather than the labels of LEX M is sufficient for computing a minimal triangulation. This mimics the simplification of using weights in MCS rather than the labels of Lex-BFS for computing a perfect elimination ordering. The new result, an algorithm that we call MCS-M, has a substantially simpler implementation that that of LEX M.

## 4 The new MCS-M algorithm

The new algorithm presented in this paper, MCS-M, is an extension of MCS in the same way that LEX M is an extension of Lex-BFS. When $v$ receives number $i$ at step $n - i + 1$, MCS-M increments the weight of all unnumbered vertices $u$ for which there exists a path between $v$ and $u$ consisting only of unnumbered vertices with weight strictly less than $w(v)$ and $w(u)$. The details of this $O(nm)$ time algorithm are given in Figure 4.

An example run of MCS-M on a 5-cycle is given in Figure 5. The figure shows the numbers and the weights of the vertices at the end of each step

**Algorithm** MCS-M
**Input:** A graph $G = (V, E)$.
**Output:** A minimal elimination ordering $\alpha$ of $G$ and the corresponding triangulated graph $H = G^+_\alpha$.
**begin**
    $F = \emptyset$;
    **for** all vertices $v$ in $G$ **do**
        $w(v) = 0$;
    **for** $i = n$ **downto** 1 **do**
        Choose an unnumbered vertex $v$ of maximum weight $w(v)$;
        $S = \emptyset$;
        **for** all unnumbered vertices $u \in G$ **do**
            **if** there is an edge $uv$ or a path $u, x_1, x_2, ..., x_k, v$ in $G$ through unnumbered vertices such that $w(x_i) < w(u)$ for $1 \leq i \leq k$ **then** $S = S \cup \{u\}$;
        **for** all vertices $u \in S$ **do**
            $w(u) = w(u) + 1$;
            **if** $uv \notin E$ **then** $F = F \cup \{uv\}$;
        $\alpha(v) = i$;
    $H = (V, E \cup F)$;
**end**

Figure 4: The MCS-M algorithm.

of the algorithm. Numbers that are assigned to vertices are shown inside each vertex, whereas the current weights of the vertices are shown next to the vertices, in parentheses. The added fill edges are given in dashed line style. Note that the added edges have no effect on the choices made by the algorithm, as the algorithm only considers edges of the original graph $G$. The output graph $H$ with the produced ordering $\alpha$ is given as the last graph of the figure.

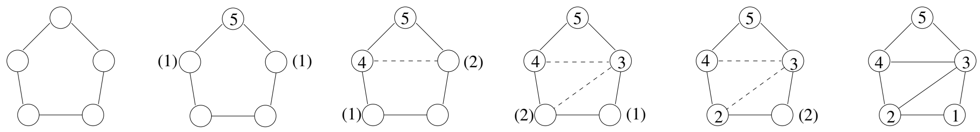

Figure 5: An example run of MCS-M on a 5-cycle.

Another execution of MCS-M is shown in Figure 6(a), and can be compared to Figure 6(b) where LEX M is applied to the same graph. In (a) and (b) final weights/labels are given in parentheses after the assigned $\alpha$ order number of each vertex. Both orderings produce the fill represented by the dashed edges in (c). The ordering is not minimum since there exists an ordering with only five fill edges. For this example it is not possible for MCS-M to produce the ordering shown in part (b), and it is not possible for LEX M to produce the ordering shown in part (a). Thus, LEX M and MCS-M do not produce the same orderings.

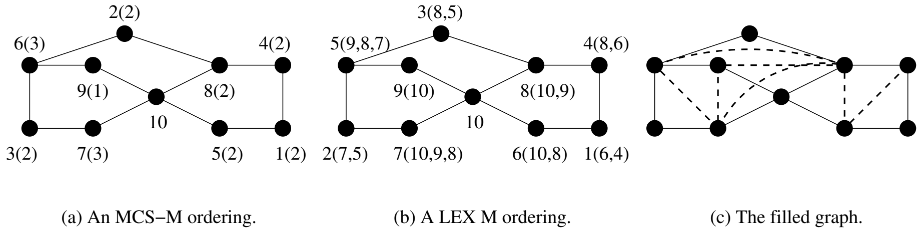


Figure 6: (a) An MCS-M numbering. (b) A LEX M numbering. (c) A minimal triangulation that is not minimum.

Throughout the remainder of this paper, while speaking about MCS or MCS-M, the following phrases are considered to be equivalent: *$u$ is numbered higher than $v$* and *$u$ is processed earlier than $v$*. The symbols $v-$ and $v+$ are used as time stamps, where $v-$ denotes the time at which $v$ is chosen as the vertex to receive its number, and $v+$ denotes the time at which $v$ receives its number. For any two vertices $u$ and $v$, where $v$ is numbered higher than $u$ during an execution of MCS or MCS-M, $w_{v-}(u)$ is the weight of $u$ at time $v-$, and $w_{v+}(u)$ is the weight of $u$ at time $v+$. Similarly, for a set of vertices $A$, $w_{v-}(A)$ and $w_{v+}(A)$ denote the *highest* weight of a vertex among the unnumbered vertices of $A \subseteq V$, at times $v-$ and $v+$, respectively.

To prove the correctness of MCS-M, we will, after establishing the fact that the computed graph $H$ is chordal, show that every fill edge is the unique chord of a 4-cycle in $H$. It will follow then by Theorem 2.2 that $H$ is a minimal triangulation of $G$. We begin by proving a property about paths containing unnumbered lower weight intermediary vertices. As with LEX M, these paths will be called *fill paths*. It will be clear from the proof of the following lemma that once such a path is established it will remain a fill path throughout the algorithm and eventually result in a fill edge between its endpoints.

**Lemma 4.1** *Let $\alpha$ be an ordering produced by an execution of MCS-M on $G$. For any step of MCS-M, let $v$ be the vertex chosen to receive its number. Among the unnumbered vertices, if*

$$w_{v-}(x_i) < w_{v-}(y) \leq w_{v-}(z)$$

*for all $x_i$ on a path $y, x_1, x_2, ..., x_r, z$ in $G$, then*

$$\alpha(x_i) < \min\{\alpha(y), \alpha(z)\}.$$

**Proof.** Suppose there is a path for which $w_{v-}(x_i) < w_{v-}(y) \leq w_{v-}(z)$ as in the premise of the lemma. Note that for any $u$ such that $\alpha(v) > \alpha(u) > \max\{\alpha(x_i), \alpha(y), \alpha(z)\}$, $w_{u-}(u) \geq \max\{w_{u-}(y), w_{u-}(z)\}$. Thus, if $w_{u-}(x_i) < \min\{w_{u-}(y), w_{u-}(z)\}$ then any lower weight path from $u$ to some

$x_i$ that causes $w_{u+}(x_i) = w_{u-}(x_i) + 1$ can be extended as lower weight paths from $u$ through $x_i$ to $y$ and to $z$, respectively, causing $w_{u+}(y) = w_{u-}(y) + 1$ and $w_{u+}(z) = w_{u-}(z) + 1$. Consequently, since MCS-M always chooses next a vertex with highest weight to receive the highest remaining number, the weights of $y$ and $z$ remain larger than the weights of all $x_i$ at later steps until $y$ or $z$ is numbered, and either $y$ or $z$ receives the highest number among the mentioned vertices. Assume without loss of generality that $\alpha(y) < \alpha(z)$. When $z$ receives its number it will increase the weight of $y$, and at all later steps until $y$ is numbered, by the same argument as above, any vertex that increases the weight of one of the vertices $x_i$ will also increase the weight of $y$. Thus, $y$ will be numbered higher than any of the $x_i$. ∎

Now, we will prove that MCS-M produces a triangulation of the input graph, by showing the stronger property that the output graph is exactly the same graph as the one that would be produced by the elimination game using the ordering $\alpha$ produced by MCS-M.

**Theorem 4.2** *Let $H$ and $\alpha$ be the graph and ordering produced by an execution of MCS-M on $G$. Then $H = G_\alpha^+$.*

**Proof.** Given an input graph $G$, let $\alpha$ be the elimination ordering and $H$ be the supergraph computed by an execution of MCS-M. In order to prove that $H = G_\alpha^+$, we will prove that a fill edge $uv$ with $\alpha(u) < \alpha(v)$ is added by MCS-M if and only if there is a path $u, x_1, x_2, ..., x_r, v$ in $G$ with $\alpha(x_i) < \alpha(u)$ for $1 \leq i \leq r$. The result will then follow from Theorem 2.1.

Let $u$ and $v$ be non-adjacent in $G$, and assume that $uv$ is added by MCS-M. This can only happen if there is a path $u, x_1, x_2, ..., x_r, v$ in $G$ where $x_i$ is unnumbered with $w_{v-}(x_i) < w_{v-}(u) \leq w_{v-}(v)$ for $1 \leq i \leq r$. Then by Lemma 4.1 $\alpha(x_i) < \alpha(u)$, for $1 \leq i \leq r$.

For the other direction, assume that there is a path $p = u, x_1, x_2, ..., x_r, v$ in $G$ with $\alpha(x_i) < \alpha(u)$ for $1 \leq i \leq r$, and assume on the contrary that $uv$ is not an edge of $H$. Let $X = \{x_1, x_2, ..., x_r\}$. Since $v$ is the first to receive its number among all mentioned vertices, $w_{v-}(v) \geq w_{v-}(u)$ and $w_{v-}(v) \geq w_{v-}(X)$. Since $uv$ is not added by MCS-M, $w_{v-}(u)$ is smaller than or equal to the highest weight on every path consisting of unnumbered vertices between $u$ and $v$ at time $v-$, and in particular, $w_{v-}(X) \geq w_{v-}(u)$. Then $w_{v+}(X) > w_{v+}(u)$, since the vertex of $X$ with the highest weight closest to $v$ on $p$ is connected to $v$ by a fill path or a direct edge, and thus gets its weight increased by $v$. Let $x_j$ be the vertex of $p$ closest to $u$ with $w_{v+}(x_j) > w_{v-}(u)$, and define $p_j = u, x_1, x_2, ..., x_{j-1}$ and $U = \{u, x_1, x_2, ...x_{j-1}\}$. Note first that $w_{v+}(x_j) > w_{v+}(U)$ since we chose $x_j$ closest to $u$ on $p$. Let now $z$ be the vertex that receives its number and increments the weight of a vertex of $U$ for the first time after time step $v+$. Then $z$ also increases the weight of $x_j$ since $w_{z-}(z) \geq w_{z-}(x_j) > w_{z-}(U)$. Therefore, $w(x_j)$ will stay higher than $w(u)$ and $w(U)$ at all later steps since every time we increase the weight of $u$ or any other vertex of $U$ we also increase $w(x_j)$. Since a vertex of highest weight is chosen at each step, this contradicts our assumption that $\alpha(x_j) < \alpha(u)$, and completes the proof. ∎

We now prove the minimality of the computed triangulation.

**Lemma 4.3** *Let $H = (V, E \cup F)$ be the triangulation of $G = (V, E)$ produced by an execution of MCS-M. Every edge belonging to $F$ is the unique chord of a 4-cycle in $H$.*

**Proof.** Observe that every edge in $F$ has at least one associated fill path that caused MCS-M to add the edge to $F$. Initially MCS-M assigns weight zero to every vertex, and thus there are no fill paths. Consequently, there are no fill paths incident with, and hence no fill edges added incident to, the first vertex chosen to be numbered $n$ by MCS-M. A fill path is then created only by increasing the weights of the endpoints of paths to values higher than the weights of the vertices interior to the path.

Let $uv$ be a fill edge generated by MCS-M and let $z$ be the vertex that first establishes a fill path between $u$ and $v$. Then there is no fill path joining $u$ and $v$ at time $z-$, but there is at least one fill path $p = u, x_1, x_2, ..., x_r, v$ at time $z+$. Without loss of generality, assume that $w_{z-}(u) \le w_{z-}(v)$. Clearly, $w_{z-}(x_i) = w_{z-}(u)$ for at least one $x_i$ on $p$, because if $w_{z-}(x_i) < w_{z-}(u)$ for every $x_i$ then $p$ would have been a fill path at time $z-$, and if $w_{z-}(x_i) > w_{z-}(u)$ for some $x_i$ then there is no way that $w_{z+}(x_i) < w_{z+}(u)$, which is required for $p$ to be a fill path at time $z+$. We will show that for some $x_i$ on $p$, where $w_{z-}(x_i) = w_{z-}(u)$, $uv$ is the unique chord in the 4-cycle $u, x_i, v, z, u$ in $H$.

We begin by establishing the existence of the edges $uz$ and $vz$ in $H$. Now, $uz$ is an edge in $H$ because the weight of $u$ must be increased by one at time $z+$ for $p$ to become a fill path. Furthermore, if $w_{z-}(u) = w_{z-}(v)$, then the same argument guarantees that $vz$ is an edge in $H$. If, on the other hand, $w_{z-}(u) < w_{z-}(v)$, we have $w_{z-}(u) < w_{z-}(v) \le w_{z-}(z)$, since $z$ is chosen at time $z-$. In this case, $w_{z-}(x_j) \le w_{z-}(u)$ for all vertices $x_j$ on $p$ and there is a fill path between $v$ and $z$ through $u$ at time $z-$. It follows that the edge $vz$ is in $H$.

Next we show that, at time $z+$, there is no edge $x_i z$ in $H$ for any $x_i$ on $p$ with $w_{z-}(x_i) = w_{z-}(u)$. Now $x_i z$ cannot be an edge of $H$, because otherwise this would mean that $z$ increased the weight of $x_i$, implying that $w_{z+}(x_i) = w_{z+}(u)$, and consequently $x_i$ would not be on a fill path between $u$ and $v$ at time $z+$.

Finally, we pick a particular $x_i$ with $w_{z-}(x_i) = w_{z-}(u)$, and show that the edges $ux_i$ and $x_i v$ are in $H$. By Lemma 4.1, all the vertices $x_j$ on $p$ for which $w_{z-}(x_j) < w_{z-}(x_i) = w_{z-}(u)$ receive numbers that are smaller than those received by any such vertex $x_i$. This means that the highest number assigned to an interior vertex on path $p$ is to such a vertex $x_i$ with

$w_{z^-}(x_i) = w_{z^-}(u)$. Let $x_i$ be the interior vertex of $p$ with the highest $\alpha$-number. By Theorem 2.1 and Lemma 4.1, $ux_i$ and $x_iv$ are edges in $H$.

We have shown that the fill edge $uv$ is the unique chord of the 4-cycle $u, x_i, v, z, u$ in $H$, giving the desired result. ∎

**Theorem 4.4** *MCS-M computes a minimal triangulation and a corresponding minimal elimination ordering.*

**Proof.** Follows from Lemma 4.3 and Theorems 2.2 and 4.2. ∎

# 5 A closer look at MCS

The results in the previous section demonstrate that maintaining cardinalities rather than lexicographic labels is sufficient for computing minimal triangulations. This naturally leads to questions about what properties a vertex numbered 1 by MCS may have in an arbitrary graph, especially in view of the interesting behavior of MCS on a chordal graph, as fully described in [6]. Given any graph $G$, we will show that a vertex of $G$ that is numbered 1 by an execution of MCS can be eliminated first in some minimal elimination ordering, as is also the case for a vertex numbered 1 by Lex-BFS (see [4]).

Ohtsuki, Cheung, and Fujisawa [13] characterized the vertices that can be numbered 1 in a minimal elimination ordering. Below we define an OCF-vertex (OCF representing the initials of the authors of [13]) as a vertex that satisfies their condition, and summarize in a theorem their results.

**Definition 5.1** *A vertex $x$ in $G = (V, E)$ is an OCF-vertex if, for each pair of non-adjacent vertices $y, z \in N(x)$, there is a path $y, x_1, x_2, ..., x_k, z$ in $G$ where $x_i \in G(V - N[x])$, for $1 \leq i \leq k$.*

**Theorem 5.2** (Ohtsuki, Cheung, and Fujisawa [13]) *A minimal elimination ordering $\alpha$ is computed by choosing, at each step $i$ of the elimination game, an OCF-vertex $x$ in $G^{i-1}$, and setting $\alpha(x) = i$.*

We now present our result, which implies that even standard MCS can be used as a tool in computing minimal triangulations.

**Theorem 5.3** *Given an arbitrary graph $G$, any vertex that can be numbered 1 by an MCS execution is an OCF-vertex of $G$.*

**Proof.** We will prove the equivalent statement that if a vertex is not an OCF-vertex then it cannot be numbered 1 by MCS. Assume on the contrary that there is a vertex $x$ in $G = (V, E)$ which is not an OCF-vertex and that an execution of MCS on $G$ gives number 1 to vertex $x$. For the remainder of this proof, we will refer to this execution.

For a vertex $y \in N(x)$, we let $C_y$ represent the union of all connected components of $G(V - N[x])$ that contain $y$ in their neighborhoods. If $C_y$ is empty then we let $N(C_y) = \{y\}$ for simplicity, and thus $C_y \cup N(C_y)$ is always non-empty. Since $x$ is not an OCF-vertex, there must exist two vertices $y, z \in N(x)$ with $yz \notin E$, such that there is no path between $y$ and $z$ with all intermediate vertices belonging to $G(V - N[x])$. Thus $C_y \cap C_z = \emptyset$. Let $Z = N(C_z) - N(C_y)$, and let $Y = N(C_y) - N(C_z)$. Thus $y \in Y$ and $z \in Z$, and the sets $Y, Z, N(C_y), N(C_z)$ are all proper subsets of $N(x)$. Let $z$ be higher numbered than $y$ by MCS.

First we make the following observation. At the step when $z$ is chosen to be the next vertex to receive its number, $w_{z-}(C_y \cup \{y\}) \geq w_{z-}(x) + 1$. For a proof of this, assume that it is not true. When $z$ receives its number it will increase $w(x)$ but not the weight of any vertex in $C_y \cup \{y\}$ since no vertex of $C_y \cup \{y\}$ is adjacent to $z$. So $w(x)$ will exceed $w(C_y \cup \{y\})$. Clearly, $w(C_y \cup \{y\})$ must eventually either equal or exceed $w(x)$ since $x$ ends up receiving number 1. So some vertex outside of $C_y \cup \{y\} \cup \{x\}$ has to be numbered before anything in $C_y \cup \{y\}$ in order to increase $w(C_y \cup \{y\})$ and obey the assumed numbering on the vertices. The important thing to note is that any such vertex must lie in $N(C_y \cup \{y\})$ and is therefore adjacent to $x$. Thus every time we try to increase $w(C_y \cup \{y\})$, we will also increase $w(x)$, which gives the desired contradiction.

From the above observation, we can deduce that $w_{z-}(x) + 1 \leq w_{z-}(C_y \cup N(C_y))$. Let $t$ be the most recent step before $z$ such that $w_t(C_y \cup N(C_y))$ was increased and therefore exceeded $w_t(x)$. Thus, at that step, a vertex $v$ received its number such that $w_{v-}(C_y \cup N(C_y)) = w_{v-}(x)$, and $w_{v+}(C_y \cup N(C_y)) = w_{v+}(x) + 1$. Because of this, and since $v$ was chosen to be the vertex to be numbered, $w_{v-}(C_y \cup N(C_y)) = w_{v-}(C_z \cup Z \cup \{x\})$, and $w_{v+}(C_y \cup N(C_y)) = w_{v+}(C_z \cup Z \cup \{x\}) + 1$. This proves the base case of the following induction hypothesis.

**Induction hypothesis:** $w_t(C_y \cup N(C_y)) > w_t(C_z \cup Z \cup \{x\})$ for $v+ \leq t \leq z-$.

Now, let the induction hypothesis be true for all steps until $t - 1 = u-$, and let us prove it for the step $t = u+$. Since it is true for $u-$, we know that the vertex $u$ chosen to receive its number belongs to $C_y \cup N(C_y)$. If $u \in C_y$, only the weight of vertices in $C_y \cup N(C_y)$ can be increased, so the induction hypothesis is true for step $u+$. If $u \in N(C_y)$, vertex $u$ might increase $w(C_z \cup Z)$ by 1, but it will definitely also increase $w(x)$ by 1 since $N(C_y) \subset N(x)$. By our assumption that $w(x)$ does not exceed $w(C_y \cup N(C_y)) - 1$ between steps $v+$ and $z-$, the induction hypothesis is true for step $u+$.

By the induction hypothesis $w_{z-}(C_y \cup N(C_y)) > w_{z-}(z)$, contradicting the assumption that $z$ is numbered at this step while $y$ is still unnumbered.

■

Theorems 5.2 and 5.3 together give a straightforward algorithm, which we call Repeated-MCS, for computing minimal triangulations: At each step $i$ of the elimination game, run MCS on $G^{i-1}$, and choose the vertex that is numbered 1 by MCS as the vertex to receive number $i$ in the resulting minimal elimination ordering. This is an $O(nm')$ time procedure, where $m'$ is the number of edges in $G^+_\alpha$.

It should be noted that MCS-M and Repeated-MCS are not equivalent. The graph shown in Figure 7 illustrates this point. The given ordering of the graph is an MCS-M ordering. However, no MCS ordering of this graph can give the lowest number to the vertex numbered 1 by MCS-M. Thus no Repeated-MCS ordering is able to pick this vertex as the first vertex to eliminate.

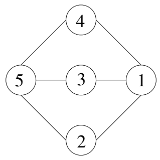


Figure 7: An example showing an MCS-M ordering which cannot be created by Repeated-MCS.

## 6 Conclusion

We have described a new algorithm, MCS-M, that computes a minimal elimination ordering and a minimal triangulation of the input graph. MCS-M can be viewed as a simplification of LEX M. As can be seen in the example of Figure 6, LEX M and MCS-M are not equivalent. The maximum label (10,8) and the maximum weight (3) just before choosing the vertex to be numbered 6 by the two algorithms force different orderings. However, one interesting open question is whether or not LEX M and MCS-M produce the same triangulations.

The time complexity of MCS-M is $O(nm)$ which follows from the time complexity analysis of LEX M given in [18], and this is the best known time for producing minimal triangulations of arbitrary graphs. In order to achieve $O(nm)$ time in [18], the authors describe an implementation of LEX M that uses label numbers, rather than lists of vertices as labels. However, in order for the label numbers to properly implement the relative lexicographic labels in LEX M, their implementation must also sort and normalize all unprocessed label numbers *after each vertex is processed.* This effectively adds a (lower-order) term to their time complexity, requiring $O(nm+n^2) = O(nm)$

time. Our MCS-M implementation does not require this extra sorting step, and has thus an easier and faster implementation.

We also leave open several other questions related to LEX M and MCS-M. How can one, for example, characterize the vertices which can be numbered 1 by each of these algorithms? How hard is it to decide whether or not a given vertex can be numbered 1? Is it possible to characterize the minimal triangulations that can be computed by LEX M and the ones that can be computed by MCS-M? For every graph $G$, is there a LEX M and/or MCS-M ordering that gives a minimum triangulation of $G$?

The most outstanding open question about minimal triangulations is whether it is possible to produce minimal triangulations of arbitrary graphs in time less than $O(nm)$.

## References


[1] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, *On the desirability of acyclic database systems*, J. Assoc. Comput. Mach., 30 (1983), pp. 479–513.

[2] A. Berry, *A wide-range efficient algorithm for minimal triangulation*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999.

[3] A. Berry, J. R. S. Blair, and P. Heggernes, *Maximum cardinality search for computing minimal triangulations*, in Graph Theoretical Concepts in Computer Science - WG 2002, L. Kucera, ed., Springer Verlag, 2002. Lecture Notes in Computer Science.

[4] A. Berry and J.-P. Bordat, *Separability generalizes Dirac's theorem*, Discrete Applied Mathematics, 84 (1998), pp. 43–53.

[5] J. R. S. Blair, P. Heggernes, and J. A. Telle, *A practical algorithm for making filled graphs minimal*, Theoretical Computer Science, 250 (2001), pp. 125–141.

[6] J. R. S. Blair and B. W. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computations, J. A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer Verlag, 1993, pp. 1–30. IMA Volumes in Mathematics and its Applications, Vol. 56.

[7] F. R. K. Chung and D. Mumford, *Chordal completions of planar graphs*, J. Comb. Theory, 31 (1994), pp. 96–106.

[8] E. Dahlhaus, *Minimal elimination ordering inside a given chordal graph*, in Graph Theoretical Concepts in Computer Science - WG '97,

R. H. Möhring, ed., Springer Verlag, 1997, pp. 132–143. Lecture Notes in Computer Science 1335.

[9] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835–855.

[10] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

[11] 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 (1988), pp. 157–224.

[12] T. Ohtsuki, *A fast algorithm for finding an optimal ordering in the vertex elimination on a graph*, SIAM J. Comput., 5 (1976), pp. 133–145.

[13] 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 (1976), pp. 622–633.

[14] S. Parter, *The use of linear graphs in Gauss elimination*, SIAM Review, 3 (1961), pp. 119–130.

[15] B. Peyton, *Minimal orderings revisited*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 271–294.

[16] D. J. Rose, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl., 32 (1970), pp. 597–609.

[17] ——, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, 1972, pp. 183–217.

[18] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[19] 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 (1984), pp. 566–579.

[20] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 77–79.