Maximum Cardinality Search for Computing Minimal Triangulations

Anne Berry[1], Jean R. S. Blair[2], and Pinar Heggernes[3]

[1] LIMOS, Universite Clermont-Ferrand II, F-63177 Aubiere, France, berry@isima.fr
[2] US Military Academy, West Point, NY, USA, Jean-Blair@usma.edu
[3] Informatics, University of Bergen, N-5020 Bergen, Norway, pinar@ii.uib.no

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 also computes a minimal triangulation of an arbitrary graph. MCS simplified the fundamental concept used in Lex-BFS, resulting in a simpler algorithm for recognizing chordal graphs. The new simpler 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

Many important problems in graph theory rely on the computation of a chordal completion or, equivalently, a triangulation of a graph. Typically the goal is to compute a *minimum* triangulation, that is, a triangulation with the fewest number of edges. Computing a minimum triangulation is NP-hard [11]. In this extended abstract, we study the problem of finding a minimal triangulation. A *minimal* triangulation H of a given graph G is a triangulation such that no subgraph of H is a triangulation of G.

Several practical algorithms exist for finding minimal triangulations [1], [2], [3], [5], [8], [9]. One such classical algorithm, called Lex-M [9], is derived from the Lex-BFS (lexicographic breadth first search) algorithm [9] for recognizing chordal graphs. 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 vertices that can be reached along special kinds of paths. Interestingly, the simple extension of adding to labels based on reachability along special kinds of paths, rather than only along single edges, results in an algorithm that produces minimal triangulations.

The adjacency-labeling concepts developed for the Lex-BFS algorithm 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

L. Kučera (Ed.): WG 2002, LNCS 2573, pp. 1–12, 2002.
© Springer-Verlag Berlin Heidelberg 2002

case of recognizing chordality, knowing the specific processed neighbors (i.e., labels) is not necessary; one need only maintain and compare the cardinality of processed neighbors [10]. This was a major breakthrough, resulting in a significantly simplified implementation of Lex-BFS which has come to be 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? Or, equivalently, can MCS be extended from neighbors to paths in order to yield a minimal triangulation algorithm, imaging 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 provably minimal triangulations of arbitrary graphs, as well as recognizing chordality. Both algorithms on the left have time complexity $O(n+m)$; both algorithms on the right have time complexity $O(nm)$.

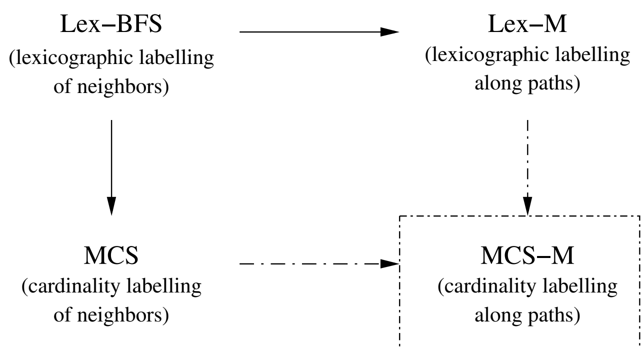

Fig. 1. Relationships between algorithms. Solid arrows represent previous evolution. Dashed arrows represent the natural evolution to a new MCS-M algorithm.

This paper is organized as follows. In the next section we assume that the reader is familiar with standard graph terminology, and briefly review only a few key definitions before presenting background material. Included in that section is a classical characterization of minimal triangulations that forms the basis for our proofs of correctness. The three algorithms that lead to the results in this paper are presented in Section 3. In Section 4 we present the new minimal triangulation algorithm MCS-M, and prove its correctness.

2 Background

All graphs in this work are undirected and finite. A graph is denoted by $G = (V, E)$, with $n \equiv |V|$, and $m \equiv |E|$. The *neighborhood* of a vertex x in G is $N_G(x) = \{y \neq x \mid xy \in E\}$. The neighborhood of a set of vertices A is $N_G(A) =$

$\cup_{x \in A} N_G(x) - A$, and we define $N_G[A] = N_G(A) \cup A$. 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 x is *simplicial* if $N(x)$ 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 ≥ 4. A *triangulation* of a graph G is a chordal graph $G^+ = (V, E \cup F)$ that results from the addition of a set F of fill edges.

Given any graph $G = (V, E)$, an *elimination ordering* α on G is simply a numbering of the vertices of G with integers from 1 to n. The algorithm shown

Algorithm EliminationGame
Input: A general graph G, and an elimination ordering α of the vertices in G.
Output: The filled graph G^+_α.
begin
  $G^0 = G$;
  for $i = 1$ **to** n **do**
    Let v be the vertex for which $\alpha(v) = i$;
    Add edges to G^{i-1} so that $N_{G^{i-1}}(v)$ becomes a clique;
    $G^i = G^{i-1} - v$;
  $G^+_\alpha = \cup_{i=0}^{n-1} G^i$;
end

Fig. 2. The elimination game.

in Figure 2, called the *elimination game*, was first introduced by Parter [7]. For any graph G and any ordering α of G, we will denote by G^k_α the transitory graph after step k of the elimination game on G. The resulting filled graph G^+_α is a triangulation of G [4]. The ordering α 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 elimination game. Fulkerson and Gross [4] showed that the class of chordal graphs is exactly the class of graphs having perfect elimination orderings.

The following theorem characterizes the edges of the filled graph.

Theorem 1. (Rose, Tarjan, and Lueker [9]) *Given a graph $G = (V, E)$ and an elimination ordering α of G, yz is an edge in G^+_α if and only if $yz \in E$ or there exists a path $y, x_1, x_2, ..., x_k, z$ in G where $\alpha(x_i) < \min\{\alpha(y), \alpha(z)\}$, for $1 \leq i \leq k$.*

Ohtsuki, Cheung, and Fujisawa [6] define α to be a *minimal elimination ordering* if G^+_α is a minimal triangulation of G and further characterize a sufficient condition for a vertex to be numbered one in a minimal elimination ordering. Below we define an OCF-vertex (OCF representing the initials of the authors of [6]) as a vertex that satisfies their condition and summarize in a theorem their results that are key in proving the correctness of our algorithm.

Definition 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 $x_i \in G - N[x]$, for $1 \leq i \leq k$.*

Theorem 2. (Ohtsuki, Cheung, and Fujisawa [6]) *A minimal elimination ordering α is computed by choosing an OCF-vertex x in G^{i-1} for elimination so that $\alpha(x) = i$, at each step i of the elimination game.*

3 Lex-BFS, Lex-M, and MCS Algorithms

The MCS algorithm, which is shown in Figure 3, is a simple linear time algorithm that processes first the vertex x for which $\alpha(x) = n$ and continues generating an elimination ordering in reverse. The MCS algorithm 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.

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

Fig. 3. Maximum Cardinality Search.

Lex-BFS has the exact same description as MCS, but uses labels that are lists of the names of the already processed neighbors instead of using weights. In the 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 computes a minimal triangulation in the following way. When v receives number i at step $n - i + 1$, it adds i to the end of the label lists of all unnumbered vertices x for which there exists a path between v and x consisting only of unnumbered vertices with lexicographically lower labels than those of v and x.

The fact that using weights rather than the labels of Lex-BFS is sufficient for computing a perfect elimination ordering was a major breakthrough, resulting in the substantially simpler implementation of MCS. In the next section we show that using weights rather than the labels of Lex-M is also sufficient for computing a minimal triangulation. This results in a substantially simpler implementation of Lex-M which we call MCS-M.

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, denoting the time right before and right after 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 $v+$. Analogously, $h_{v-}(A)$ and $h_{v+}(A)$ denote the highest weight of a vertex among the unnumbered vertices of $A \subseteq V$, at times $v-$ and $v+$, respectively.

4 The New MCS-M Algorithm

The new algorithm MCS-M is an extension of MCS in the same way that Lex-M is an extension of Lex-BFS. That is, in MCS-M when v receives number i at step $n - i + 1$, it increments the weight of all unnumbered vertices x for which there exists a path between v and x consisting only of unnumbered vertices with weight strictly less than $w_{v-}(v)$ and $w_{v-}(x)$. The details of this $O(nm)$ time algorithm are given in Figure 4. An example of an MCS-M ordering on a given graph is shown in Figure 5(a).

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

Fig. 4. The MCS-M algorithm.

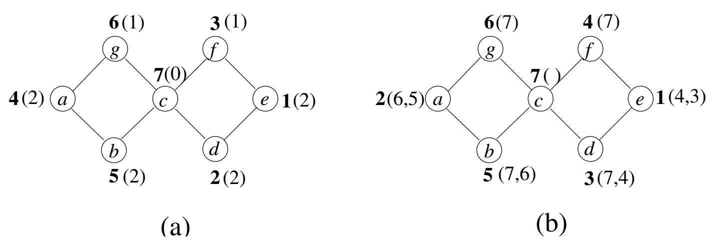

Fig. 5. (a) An MCS-M numbering. (b) A Lex-M numbering. Numbers in bold represent the produced ordering α. The weight/label of each vertex at the time it receives its number is given in parentheses.

We will show that MCS-M simulates a process of choosing an OCF vertex at each step of the elimination game, thereby producing a minimal triangulation. We begin by proving a property about paths with lower weight intermediary vertices, after which we prove that MCS-M produces exactly the same graph as the one that would be produced by the elimination game using the ordering α produced by MCS-M.

Lemma 1. *Let α be an elimination 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 $\alpha(v) > \alpha(u) > \min\{\alpha(x_i), \alpha(y), \alpha(z)\}$, $w_{u-}(u) \geq \max\{w_{u-}(y), w_{u-}(z)\}$. 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 a lower weight path through x_i to y and z causing $w_{u+}(y) = w_{u-}(y) + 1$ and $w_{u+}(z) = w_{u-}(z) + 1$. Since MCS-M always chooses next a vertex with highest weight to receive the highest remaining number, the result follows by induction. ∎

Theorem 3. *Let H and α 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 α 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 yz with $\alpha(y) < \alpha(z)$ is added by MCS-M if and only if there is a path $y, x_1, x_2, ..., x_r, z$ in G with $\alpha(x_i) < \alpha(y)$ for $1 \leq i \leq r$. The result will then follow from Theorem 1. **(⇒)** Since yz is added, there is a path $y, x_1, x_2, ..., x_r, z$ in G where x_i is unnumbered with $w_{z-}(x_i) < w_{z-}(y) \leq w_{z-}(z)$ for $1 \leq i \leq r$. Then by Lemma 1 $\alpha(x_i) < \alpha(y)$, for $1 \leq i \leq r$. **(⇐)** Let $X = \{x_1, x_2, ..., x_r\}$. Since z is the first to receive its number among all mentioned vertices, $w_{z-}(z) \geq w_{z-}(y)$ and $w_{z-}(z) \geq h_{z-}(X)$. We want to prove that $h_{z-}(X) < w_{z-}(y)$, which means that $w(y)$ is incremented and yz is added when z receives its number. Assume on the contrary that $h_{z-}(X) \geq w_{z-}(y)$ and that yz is not added. Then $h_{z+}(X) > w_{z+}(y)$. Let j be the index such that $x_j \in X$ is the closest to y among vertices of X with $w_{z+}(x_j) > w_{z+}(y)$. When a vertex q receives its number and increments $w(y)$ for the first time after the numbering of z, it will also increment $w(x_j)$ since y is on the path between x_j and q and has lower weight. Thus we cannot increment $w(y)$ without incrementing $w(x_j)$, which contradicts that $\alpha(y) > \alpha(x_j)$. ∎

We have shown that the filled graph produced by MCS-M is equivalent to the graph produced by the elimination game using the same ordering. In proving our main lemma (Lemma 4), we will use this to infer the existence of fill edges added during MCS-M, which in turn implies the existence of paths in G through lower numbered vertices. First we prove two other necessary results.

Lemma 2. *Let α be an elimination 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 for all u with $\alpha(u) > \alpha(v)$,
$$w_{u-}(x_i) \leq \min\{w_{u-}(y), w_{u-}(z)\}.$$

Proof. Let v and the path $y, x_1, x_2, ..., x_r, z$ in G be as stated in the premise and suppose to the contrary that for some vertex u with $\alpha(u) > \alpha(v)$, there exists a vertex x_i on the path for which $w_{u-}(x_i) > \min\{w_{u-}(y), w_{u-}(z)\}$. Without loss of generality assume $w_{u-}(y) \leq w_{u-}(z)$ and let u and x_i be such that x_i is the the closest vertex to y on the path that has $w_{u-}(x_i) > w_{u-}(y)$ at some time before v is numbered. Let p_x be the portion of the path between y and x_i. Thus, we have $w_{u-}(x_i) > w_{u-}(y) \geq w_{u-}(x_j)$ for all x_j on p_x. Since $w_{v-}(y) > w_{v-}(x_i)$ for the later (lower) numbered vertex v, there must be a vertex q, $\alpha(u) > \alpha(q) > \alpha(v)$, such that $w_{q-}(x_j) \leq w_{q-}(y) < w_{q-}(x_i)$ and $w_{q+}(y) = w_{q+}(x_i)$. But this cannot happen for the following reasons. The fact that q is the next to be numbered vertex means that $w_{q-}(q) \geq w_{q-}(x_i)$. The increase of y when q is numbered means there is a path (possibly a single edge), say p_1, between q and y that allowed the weight of y to be increased. The path $q - p_1 - y - p_x - x_i$ then is a lower weight path between q and x_i that would result in the weight of x_i being incremented as well, contradicting the assumption that the weight of y and not x_i is increased when q is numbered. ■

Lemma 3. *Let α be an elimination ordering produced by an execution of MCS-M and consider the vertices u and v_1, $\alpha(u) < \alpha(v_1)$, such that MCS-M increments $w(u)$ through a path (or single edge) $p_v = v_1, v_2, \cdots, v_r, u$ of lower weight intermediate vertices when processing v_1. Let x be any vertex with $\alpha(x) < \alpha(u)$ and define $k = \alpha(x)$. If $w_{v_1-}(x) = w_{v_1-}(u)$ and $v_i x$ is an edge in G_α^{k-1} for some $1 \leq i \leq r$ then MCS-M also increments $w(x)$ when processing v_1.*

Proof. Assume $w_{v_1-}(x) = w_{v_1-}(u)$ and $v_i x$ is an edge in G_α^{k-1}. Either xv_i is an edge in G or it is a fill edge introduced when v_i is numbered by MCS-M. In either case, there is a path p_{small} (or single edge) connecting x and v_i in G such that $h_{v_i-}(p_{small}) < w_{v_i-}(x) \leq w_{v_i-}(v_i)$. Applying Lemma 2 we see that

$$\begin{aligned} h_{v_1-}(p_{small}) &\leq \min\{w_{v_1-}(x), w_{v_1-}(v_1)\} \\ &\leq w_{v_1-}(v_1) \\ &< w_{v_1-}(u) \ \ \text{(by the definition of } p_v) \\ &= w_{v_1-}(x) \end{aligned}$$

It follows that $v_2, \cdots, v_i - p_{small}$ is a lower weight path through unnumbered vertices connecting v_1 and x just before v_1 is processed by MCS-M. Thus, $w_{u+}(x) = w_{u-}(x) + 1$. ■

Lemma 4. *Let α be an elimination ordering produced by an execution of MCS-M. For $1 \leq k \leq n$, if $\alpha(y) = k$ then y is an OCF vertex in G_α^{k-1}.*

Proof. Let α be an elimination ordering produced by an execution of MCS-M, and consider a vertex y_0 with $\alpha(y_0) = k$. Let y_1 and y_2 be any two vertices in $N_{G_\alpha^{k-1}}(y_0)$ with $y_1 y_2 \notin E(G_\alpha^{k-1})$. We will show that there exists a path p_h between y_1 and y_2 in G_α^{k-1} with all intermediate vertices belonging to $G_\alpha^{k-1} - N_{G_\alpha^{k-1}}[y_0]$, thereby proving that y_0 is an OCF vertex in G_α^{k-1}.

Without loss of generality, assume $\alpha(y_1) < \alpha(y_2)$ and hence that $\alpha(y_0) < \alpha(y_1) < \alpha(y_2)$. Since $y_0 y_1$ is an edge in G_α^{k-1}, either $y_0 y_1$ is in G or it is introduced by MCS-M when y_1 is processed. In either case, at time y_1- there is a path (or possibly an edge) $p_{y_0y_1}$ in G through unnumbered vertices such that $h_{y_1-}(p_{y_0y_1}) < w_{y_1-}(y_0) \leq w_{y_1-}(y_1)$. Likewise at time y_2- there is a path $p_{y_0y_2}$ through unnumbered vertices such that $h_{y_2-}(p_{y_0y_2}) < w_{y_2-}(y_0) \leq w_{y_2-}(y_2)$.

The fill edge $y_1y_2 \in G_\alpha^+$ because it is introduced during the elimination game by y_0. It follows then from Theorem 3 that the edge y_1y_2 is introduced by MCS-M when y_2 is numbered. Hence there is a path $y_1, v_1, v_2, \cdots, v_r, y_2$, $r \geq 1$, such that $w_{y_2-}(v_i) < w_{y_2-}(y_1) \leq w_{y_2-}(y_2)$, for all $1 \leq i \leq r$. If $w_{y_2-}(y_0) < w_{y_2-}(y_1)$, then the path $p_{y_0y_1} - y_0 - p_{y_0y_2}$ provides such a path. We consider first, however, the case where $w_{y_2-}(y_0) \geq w_{y_2-}(y_1)$.

Observe that since $y_1y_2 \notin G_\alpha^{k-1}$, there is at least one vertex on $p_{alt} = v_1, v_2, \cdots v_r$ that is higher numbered then y_0. We will show that the vertices on p_{alt} that are higher numbered than y_0 form the desired path p_h in G_α^{k-1}. By Theorem 1 the vertices on p_{alt} that are higher numbered than y_0 induce a path in G_α^{k-1} between y_1 and y_2. Thus, we need only show that no vertex on p_{alt} is adjacent to y_0 in G_α^{k-1}.

Assume to the contrary that there is a vertex v_i on p_{alt} that is adjacent to y_0 in G_α^{k-1}. Either y_0v_i is an edge in G or it is a fill edge introduced when v_i is numbered by MCS-M. In either case, there is a path (or edge) p_{small} connecting y_0 and v_i in G such that $h_{v_i-}(p_{small}) < w_{v_i-}(y_0) \leq w_{v_i-}(v_i)$. Applying Lemma 2 we see that $h_{y_2-}(p_{small}) \leq \min\{w_{y_2-}(y_0), w_{y_2-}(v_i)\} \leq w_{y_2-}(v_i)$. We further know that $w_{y_2-}(v_i) < w_{y_2-}(y_1) \leq w_{y_2-}(y_0)$, since p_{alt} is the path through which MCS-M added the edge y_1y_2. Therefore $h_{y_2-}(p_{small}) < w_{y_2-}(y_0)$. This gives us two paths in G:

$$y_0 - p_{small} - v_i, v_{i-1}, ..., v_1, y_1$$

and

$$y_0 - p_{small} - v_i, v_{i+1}, ..., v_r, y_2$$

that, just before y_2 is numbered by MCS-M, satisfy the premise to Lemma 1. Combined the two paths contain all of the vertices of p_{alt} as internal vertices. Thus, by Lemma 1 we can conclude that every vertex v_i on p_{alt} is such that $\alpha(v_i) < \alpha(y_0)$, contradicting the fact that at least one vertex on p_{alt} is numbered higher than y_0. It follows that our assumption that v_i is adjacent to y_0 in G_α^{k-1} was wrong, and therefore that the path p_h of vertices on p_{alt} that are higher numbered than y_0 is a path in G_α^{k-1} from y_1 to y_2 through vertices that are not adjacent to y_0.

We are left then with the case where the path $p_{y_0y_1} - y_0 - p_{y_0y_2}$ is a path of lower weight vertices between y_2 and y_1 just before y_2 is numbered, and hence $w_{y_2-}(y_0) < w_{y_2-}(y_1)$. In this case we know that there is some vertex, say y_3 that first (earliest in MCS-M) increases the weight of y_1 to a value greater than the weight of y_0.

The path p_h will be constructed iteratively from its two endpoints, y_1 and y_2, towards its center, through y_3 and vertices like it. That is, we will be growing two subpaths, p_{odd} from y_1 and p_{even} from y_2, that will eventually meet to form p_h. Simultaneously we will show by induction that the vertices on p_{odd} and p_{even} are not adjacent to y_0 in G_α^{k-1}. In order to prove this, we will utilize properties of another path that goes through y_0 and overlaps with portions of p_{odd} and p_{even}.

At odd steps of the induction the subpath p_{odd} is extended from its endpoint that is furtherest away from y_1; at even steps of the induction the subpath p_{even} is extended from its endpoint that is furtherest away from y_2. If y_{i-2} is the current to-be-extended endpoint, then the corresponding subpath is extended through a particular path to a vertex y_i. These extensions are defined as follows. Let y_i be the first vertex for which $w_{y_i+}(y_{i-2}) > w_{y_i+}(y_{i-3})$. Note then that $w_{y_i-}(y_{i-2}) = w_{y_i-}(y_{i-3})$, and $w_{y_i+}(y_{i-2}) = w_{y_i-}(y_{i-2}) + 1$. Define $p_{y_i y_{i-2}}$ to be the path of lower weight unnumbered vertices at time y_i- through which MCS-M increments the weight of y_{i-2}. The partial path of p_h is then defined recursively as follows.

$$p_h^i = \begin{cases} \emptyset & \text{if } i = 2 \\ p_h^{i-1} \text{ extended to include } p_{y_i y_{i-2}} \text{ and } y_i & \text{if } i > 2 \end{cases}$$

The overlapping path that goes through y_0 is defined recursively as follows.

$$p_0^i = \begin{cases} p_{y_0 y_1} - y_0 - p_{y_0 y_2} & \text{if } i = 2 \\ p_0^{i-1} \text{ extended to include } p_{y_i y_{i-2}} \text{ and } y_{i-2} & \text{if } i > 2 \end{cases}$$

The path p_0^5 and the corresponding p_h^5 are shown in Figure 6. Note that for $i > 2$ the path p_0^i contains all of the partial path p_h^i except for its two internal endpoints.

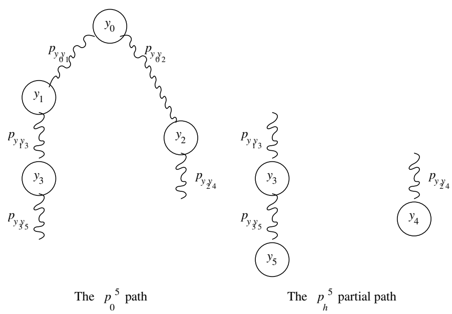

Fig. 6. The path p_0^5 and the partial path p_h^5.

There are four properties, shown in the induction hypothesis below, that we will maintain throughout the induction. The second and third are properties of the p_0 path and the last is the desired property of the p_h subpaths.

Induction hypotheses: For all l, $2 \leq l < i$, the following properties hold:

α-ORDER: $\alpha(y_{l-1}) < \alpha(y_l) < \alpha(y_{l+1})$.
SAME-WEIGHT: $w_t(y_0) = w_t(y_j)$ for $1 \leq j \leq l-1$ at times $t = y_{l+1}-$ and earlier.
p_0-WEIGHT: $h_t(p_0^l) = w_t(y_0) \leq \min\{w_t(y_l), w_t(y_{l-1})\}$ at times $t = y_l-$ and earlier.
NO-y_0-ADJ: No vertex on p_h^l is adjacent to y_0 in G_α^{k-1}.

Base case ($i = 2$): Here we begin with the fact that $\alpha(y_0) < \alpha(y_1) < \alpha(y_2)$ and observe that $\alpha(y_3) > \alpha(y_2)$, since at the time that y_2 is processed by MCS-M the weight of y_1 is already higher than the weight of y_0. Thus, the α-ORDER property holds for the base case.

The SAME-WEIGHT property trivially holds since, by definition of y_3, $w_t(y_0) = w_t(y_1)$ at all times $t = y_3-$ and earlier.

For the p_0-WEIGHT property recall that when $p_{y_0y_1}$ and $p_{y_0y_2}$ were defined above we saw that $h_{y_1-}(p_{y_0y_1}) < w_{y_1-}(y_0) \leq w_{y_1-}(y_1)$ and $h_{y_2-}(p_{y_0y_2}) < w_{y_2-}(y_0) \leq w_{y_2-}(y_2)$. Combining these two facts with the fact that $w_{y_2-}(y_0) < w_{y_2-}(y_1) \leq w_{y_2-}(y_2)$ and applying Lemma 2, we see that $h_t(p_{y_0y_1} - y_0 - p_{y_0y_2}) \leq \min\{w_t(y_1), w_t(y_2)\}$ at all times $t = y_2-$ and earlier, giving the base case p_0-WEIGHT property.

The NO-y_0-ADJ property holds since at this first step in the iteration the path p_h^2 is empty, and hence has no adjacency to y_0 in G_α^{k-1}.

Induction step ($i > 2$): Assume the induction hypotheses hold. We must either close the path p_h at this step, or prove the four properties hold for the i^{th} iteration. We begin by establishing that $w_{y_i-}(y_{i-2}) \leq w_{y_i-}(y_{i-1})$. By definition of y_i,

$$\begin{aligned} w_{y_i-}(y_{i-2}) &= w_{y_i-}(y_{i-3}) \\ &\leq h_{y_i-}(p_0^{i-1}) \quad \text{(because } y_{i-3} \in p_0^i\text{)} \\ &\leq \min\{w_{y_i-}(y_{i-1}), w_{y_i-}(y_{i-2})\} \quad \text{(by the induction hypothesis)} \end{aligned}$$

Thus, $w_{y_i-}(y_{i-2}) \leq w_{y_i-}(y_{i-1})$, and we have two cases.

Case 1 ($w_{y_i-}(y_{i-2}) = w_{y_i-}(y_{i-1})$): In this case y_i must also increment the weight of y_{i-1} through a path p_{alt} not containing y_0. To see this, observe first that it cannot increment the weight of y_{i-1} using a path containing y_0 since, by the SAME-WEIGHT induction hypothesis, $w_{y_i-}(y_0) = w_{y_i-}(y_{i-2}) = w_{y_i-}(y_{i-1})$ and thus y_0 cannot be on a lower weight path between y_i and y_{i-1}. Furthermore, if the weight of y_{i-1} were not incremented when y_i was processed, then $w_{y_i+}(y_{i-2}) > w_{y_i+}(y_{i-1})$. Since the MCS-M processing time y_i+ is no later than $y_{i-1}-$, we know also from the p_0-WEIGHT induction hypothesis that

$$\begin{aligned} h_{y_i+}(p_0^{i-1}) &\leq \min\{w_{y_i+}(y_{i-2}), w_{y_i+}(y_{i-1})\} \\ &\leq w_{y_i+}(y_{i-1}) \\ &< w_{y_i+}(y_{i-2}) \end{aligned}$$

Therefore, at any time after y_i+ that the weight of y_{i-1} is incremented, the lower weight path (or edge) that was used to increment the weight of y_{i-1} can be extended through p_0^{i-1} as a lower weight path to increment the weight of y_{i-2}. But then the weight of y_{i-2} will always exceed the weight of y_{i-1}, contradicting the α-ORDER induction hypothesis.

Now consider this path p_{alt} of lower weight vertices connecting y_i and y_{i-1} at the time that y_i is processed. By Lemma 3 ($x = y_0$, $u = y_{i-1}$, $v_1 = y_i$ and $v_2, \cdots v_r = p_{alt}$) the vertices on p_{alt} are not adjacent to y_0 in G_α^{k-1}. Combining this with the NO-y_0-ADJ induction hypothesis, we see that the vertices on the p_h^{i-1} connected together through $p_{y_iy_{i-2}} - y_i - p_{alt}$ that are higher numbered than y_0 form the desired path p_h in G_α^{k-1} that is not adjacent to y_0 in G_α^{k-1}.

Case 2 ($w_{y_i-}(y_{i-2}) < w_{y_i-}(y_{i-1})$): For this case we prove that the four properties hold for the next iteration in constructing p_h. We begin with the p_0-WEIGHT property and observe that by definition, $h_{y_i-}(p_{y_iy_{i-2}}) < w_{y_i-}(y_{i-2})$. Also, by the p_0-WEIGHT induction hypothesis, $h_{y_i-}(p_0^{i-1}) \leq w_{y_i-}(y_{i-2})$. Thus, since $w_{y_i-}(y_{i-2}) < w_{y_i-}(y_{i-1})$, we have $h_{y_i-}(p_{y_iy_{i-2}} - y_{i-2} - p_0^{i-1}) < w_{y_i-}(y_{i-1}) \leq$

$w_{y_i-}(y_i)$. And then by Lemma 2 $h_t(p_{y_i y_{i-2}} - y_{i-2} - p_0^{i-1}) \leq \min\{w_t(y_i), w_t(y_{i-1})\}$ at all times $t = y_i-$ and earlier, proving the p_0-WEIGHT property.

Note that since $w_{y_i-}(y_{i-2}) < w_{y_i-}(y_{i-1})$ there exists a vertex y_{i+1} that increases the weight of y_{i-1} beyond the weight of y_{i-2} for the first time. Furthermore, $\alpha(y_i) < \alpha(y_{i+1})$ since at time y_i- the vertex y_{i+1} had already increased the weight of y_{i-1} past the weight of y_{i-2}. This proves the next α-ORDER property.

The next NO-y_0-ADJ property comes from Lemma 3 where $x = y_0$, $u = y_{i-2}$, $v_1 = y_i$ and $v_2, \cdots v_r = p_{y_i y_{i-2}}$.

By the definition and existence of y_{i+1} we know that $w_t(y_{i-1}) = w_t(y_{i-2})$ at times $t = y_{i+1}-$ and earlier. This, together with the SAME-WEIGHT induction hypothesis, gives us the SAME-WEIGHT property for the next iteration.

We have proven by induction that at each step in the iterative process either the path p_h is completed or there exists an extension to one of the subpaths of p_h that is being constructed. Since there are a finite number of vertices in the graph G, the iteration process must eventually not be able to extend a subpath of p_h and hence, the path p_h must be completed. It follows then, that the vertex y_0 is an OCF-vertex in G_α^{k-1}. ∎

Theorem 4. *MCS-M computes a minimal triangulation.*

Proof. Follows from Lemma 4 and Theorem 2. ∎

5 Conclusion

We have described a new algorithm MCS-M that computes a minimal elimination ordering and a minimal triangulation of a graph. MCS-M can be viewed as a simplification of the Lex-M algorithm for computing a minimal triangulation. In fact, in [9] a clever implementation of Lex-M is described that uses label numbers, rather than lists of vertices as labels. The storage used and comparisons made in that implementation are similar to those required with the use of weights in MCS-M. However, in order for the label numbers to properly implement the relative lexicographic labels in Lex-M, their implementation must 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, thereby avoiding the extra term in the time complexity.

As can be seen in the example of Figure 5, Lex-M and MCS-M are not equivalent; the MCS-M ordering shown in Figure 5(a) cannot be produced by Lex-M, and the Lex-M ordering shown in Figure 5(b) cannot be produced by MCS-M.

The impetus for generating minimal triangulations is the desire to approximate minimum triangulations, since in general finding minimum triangulations is NP-hard. It is well know that minimal triangulations can have substantially more fill edges than minimum triangulations. In many cases, MCS-M can produce triangulations with less than half the fill of other minimal triangulations.

But like Lex-M, MCS-M cannot always do so well. For example, for the graph representing an $n \times n$ square grid, MCS-M produces exactly the same fill as Lex-M does, and as pointed out in [9], the minimum fill for such graphs is $O(n^2 \log n)$ whereas the fill produced by Lex-M (and hence MCS-M) is $O(n^3)$.

References

1. A. Berry, *A wide-range efficient algorithm for minimal triangulation*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999.
2. 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.
3. 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.
4. D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835–855.
5. 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.
6. 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.
7. S. Parter, *The use of linear graphs in Gauss elimination*, SIAM Review, 3 (1961), pp. 119–130.
8. B. Peyton, *Minimal orderings revisited*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 271–294.
9. 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.
10. 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.
11. M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 77-79.