A Vertex Incremental Approach for Dynamically Maintaining Chordal Graphs

Anne Berry[1], Pinar Heggernes[2], and Yngve Villanger[2]

[1] ISIMA, Univ Clermont-Ferrand II, 63177 Aubiere, France. berry@isima.fr
[2] Informatics, Univ of Bergen, 5020 Bergen, Norway. {pinar, yngvev}@ii.uib.no

Abstract. For a chordal graph G, we study the problem of whether a new vertex u and a given set of edges between u and vertices of G can be added to G so that the resulting graph remains chordal. We show how to resolve this efficiently, and at the same time, if the answer is no, define a maximal subset of the proposed edges that can be added, or conversely a minimal set of extra edges that should be added in addition to the given set. Based on these results, we present a new algorithm which computes both a minimal triangulation and a maximal chordal subgraph of an arbitrary input graph in $O(nm)$ time. This time complexity matches the best known time bound for minimal triangulation, using a totally new vertex incremental approach. In opposition to previous algorithms, our process adds each new vertex without reconsidering any choice made at previous steps, and without requiring any knowledge of the vertices that might be added at further steps.

1 Introduction

Chordal graphs are a well studied class, with applications in many fields. One important aspect in applications is maintaining a chordal graph incrementally, and previous work has dealt with the problem of adding or removing an arbitrary *edge* while maintaining chordality [6], [16].

When the graph fails to be chordal, edges can be added or removed to obtain a chordal graph: either add edges until the graph becomes chordal, a process called *triangulation*, or remove edges until the graph becomes chordal, thus computing a *chordal subgraph*. Adding or removing a minimum number of edges has been shown to be NP-hard [17], [22]. However, adding or removing an inclusion minimal set of edges can be done in polynomial time. Given an arbitrary chordal subgraph (e.g., an independent set on the vertices of the graph) or supergraph (e.g, a complete graph on the same vertex set) of the input graph, edges can be added or removed one by one after testing that the resulting graph remains chordal, until no further candidate edge can be found. This ensures that minimality is achieved, by the results of [18]. The problem of maintaining a chordal graph by edge addition or deletion and the problem of computing a maximal chordal subgraph or a minimal chordal supergraph are thus strongly related.

The problem of adding an inclusion minimal set of *fill* edges, called *minimal triangulation*, has been well studied since 1976, and several $O(nm)$ time

T. Ibaraki, N. Katoh, and H. Ono (Eds.): ISAAC 2003, LNCS 2906, pp. 47–57, 2003.

algorithms exist for solving it [3], [4], [10], [18], though none of these algorithms uses an edge incremental approach as described above. However, the algorithm proposed in [7], which requires even less time when the fill is small, does use an edge deletion approach. The reverse problem of computing a maximal chordal subgraph has also been studied, and several $O(\Delta m)$ time algorithms exist, where Δ is the maximum degree in the graph [2], [11], [21].

In this paper, we present a new process for adding a *vertex* with a given set of incident edges to a chordal graph while maintaining chordality, which we are able to implement more efficiently than if we were to add the corresponding edges one by one. Our process is based on two new characterizations. The first is a characterization of a chordal graph by its edges, and the second is a characterization of the set of edges R incident to a vertex u which must be added to a chordal graph along with edge (u, v) to ensure that chordality is preserved. We show that we can compute this set R of edges in $O(n)$ time, by proposing a data structure which corresponds to a clique tree of the current chordal subgraph.

We use our results to compute both a minimal triangulation and a maximal chordal subgraph of a given arbitrary graph in $O(nm)$ time. This is done by an incremental process which repeatedly adds a new vertex u to the already constructed chordal graph H along with a maximal set of edges of the input graph between u and H, or a minimal set of extra edges between u and H in addition to original such edges.

Some of the existing algorithms which compute a maximal chordal subgraph or a minimal triangulation also use a vertex incremental process [2], [4], [5], [11], [18], [21], though none of them compute both chordal graphs at the same time. In addition, all these previous algorithms require knowing the whole graph in advance, as either vertices that are not yet processed are marked in some way to define the next vertex in the process, or edges are added between pairs of vertices that are not yet processed.

Our approach here is completely different from the previous ones, as it is more general: At each vertex addition step, we do not require the added vertex to be or to become simplicial, which enables us to process the vertices in any order. Moreover, we add only edges incident to the new vertex, so that we never need to reconsider or change the chordal graph which has been computed so far.

As a result, our process can add any vertex with any proposed neighborhood, and efficiently give a correction if the resulting graph fails to be chordal, either by computing a maximal subset of the edges to be added, or a minimal set of extra edges along with the proposed ones. In addition, the transitory chordal graph is maintained in a dynamic fashion, as making the desired or necessary additions to the graph does not require a recomputation from start, which would be the case for the other mentioned algorithms if the input graph was to be extended with new vertices after some steps or the end of the computation.

2 Graph Theoretic Background and Notation

We assume that all input graphs are simple and connected. For disconnected graphs, the results can be applied on each connected component. A graph is denoted $G = (V, E)$, with $n = |V|$, and $m = |E|$. A vertex sequence $v_1 - v_2 - \ldots - v_k$ describes a *path* if (v_i, v_{i+1}) is an edge for $1 \leq i < k$. The *length* of a path is the number of edges in it. A *cycle* is a path that starts and ends with the same vertex. A *chord* of a cycle (path) is an edge connecting two non-consecutive vertices of the cycle (path). A *clique* is a set of vertices that are all pairwise adjacent. A *simplicial* vertex is one whose neighborhood induces a clique.

For the following definitions, we will omit subscript G when the graph is clear from the context. The *neighborhood* of a vertex v in G is $N_G(v) = \{u \neq v \mid (u, v) \in E\}$, and for a set of vertices A, $N_G(A) = \cup_{x \in A} N_G(x) - A$. $G(A)$ is the subgraph induced by a vertex set $A \subseteq V$, but we often denote it simply by A when there is no ambiguity. We would like to stress that when we say merely *subgraph* we do not necessarily mean an induced subgraph. Thus subgraph can be a proper subgraph on the same vertex set with fewer edges.

A subset S of V is called a *separator* if $G(V \setminus S)$ is disconnected. S is a *uv-separator* if vertices u and v are in different connected components of $G(V \setminus S)$, and a *minimal uv-separator* if no subset of S is a uv-separator. S is a *minimal separator* of G if there is some pair $\{u, v\}$ of vertices in G such that S is a minimal uv-separator. Equivalently, S is a minimal separator if there exist two connected components C_1 and C_2 of $G(V \setminus S)$ such that $N_G(C_1) = N_G(C_2) = S$.

A pair of non adjacent vertices $\{u, v\}$ is a *2-pair* in G if there is no chordless path of length 3 or more between u and v [14]. If G is not connected, then two vertices that belong to different connected components constitute a 2-pair by definition. If G is connected, it has been shown that $\{u, v\}$ is a 2-pair iff $N(u) \cap N(v)$ is a minimal uv-separator of G [1], [19].

A graph is *chordal* if it contains no chordless cycle of length ≥ 4. Consequently, all induced subgraphs of a chordal graph are also chordal. G is chordal iff every minimal separator of G is a clique [12]. Chordal graphs are the intersection graphs of subtrees of a tree [9], [13], [20], and the following result on this graph class gives a very useful tool which we will use as a data structure in our algorithm.

Theorem 1. ([9], [13], [20]) *A graph G is chordal iff there exists a tree T, whose vertex set is the set of maximal cliques of G, that satisfies the following property: for every vertex v in G, the set of maximal cliques containing v induces a connected subtree of T.*

Such a tree is called a *clique tree* [8], and each tree node of T is a vertex set of G corresponding to a maximal clique of G. We will not distinguish between cliques of G and their corresponding tree nodes. In addition, it is customary to let each edge (K_i, K_j) of T hold the vertices of $K_i \cap K_j$. Thus edges of T are also vertex sets. Although a chordal graph can have many different clique trees, these all share the following important properties that are related to an efficient implementation of our algorithm.

Theorem 2. ([9], [15]) *Let T be a clique tree of a chordal graph G. Every edge of T is a minimal separator of G, and for every minimal separator S in G, there is an edge $(K_i, K_j) = K_i \cap K_j = S$ in T.*

Theorem 3. ([8]) *T is a clique tree of G iff for every pair of distinct cliques K_i and K_j in G, the intersection $K_i \cap K_j$ is contained in every node of T (maximal clique of G) appearing on the path between K_i and K_j in T.*

Note that as a consequence, the intersection $K_i \cap K_j$ is also contained in every edge of T (minimal separator of G) appearing on the path between K_i and K_j in T. A chordal graph has at most n maximal cliques and $n-1$ minimal separators, and hence the number of nodes and edges in a clique tree is $O(n)$.

3 A New Characterization of Chordal Graphs

In this section we present a new characterization of chordal graphs that will be the basis of our algorithm.

Definition 1. *We will say that an edge (u, v) is* mono saturating *in $G = (V, E)$ if $\{u, v\}$ is a 2-pair in $G' = (V,\ E \setminus \{(u, v)\})$.*

Theorem 4. *A graph is chordal iff every edge is mono saturating.*

Proof. Let $G = (V, E)$ be chordal, and assume on the contrary that there is an edge $(u, v) \in E$ which is not mono saturating. Thus G' is connected, and $N(u) \cap N(v)$ is not a uv-separator in G'. Let us remove $N(u) \cap N(v)$ from G'. There is still a path connecting u and v in the remaining graph. Let p be a shortest such path. Now, p contains a vertex $x \in N(u)$ which is not adjacent to v, and a vertex $z \in N(v)$ which is not adjacent to u. Thus the following is a chordless cycle of length at least 4 in G: $u-p-v-u = u-x-...-z-v-u$, which contradicts our assumption that G is chordal. For the other direction, let every edge in G be mono saturating, and assume on the contrary that G is not chordal. Thus there exists a chordless cycle C of length at least 4 in G. Let (u, v) be any edge of C. Since at least one other vertex of C must be removed to disconnect u and v in G', any minimal uv-separator of G' contains a vertex x of C, where $x \notin N(u)$ or $x \notin N(v)$. Therefore $N(u) \cap N(v)$ cannot be a uv-separator, which contradicts our assumption that every edge is mono saturating.

Corollary 1. *Given a chordal graph $G = (V, E)$, where $(u, v) \notin E$, the graph $(V,\ E \cup \{(u, v)\})$ is chordal iff $\{u, v\}$ is a 2-pair in G.*

As a consequence, while maintaining a chordal graph by adding edges, we could check every edge of the input graph to see if the endpoints constitute a 2-pair in the transitory chordal subgraph. However, this approach requires that we check every edge several times, as pairs of vertices can become 2-pairs only after the addition of some other edges. Our main result, to be presented as Theorem

5, gives a more powerful tool that allows examining each edge of the input graph only *once* during such a process, which yields our interesting time complexity.

Assume the following scenario: we are given a chordal graph G, and we want to add an edge (u, v) to G; since we want the resulting graph to remain chordal, we must allow addition of other necessary edges to achieve this, but we allow only addition of edges incident to u. (Vertex u is the most recently added vertex in the vertex incremental approach described in the next section.) Naturally, if we add every edge between u and the other vertices of G, the resulting graph is chordal. However, our main goal is to add as few edges as possible. Theorem 5 gives a necessary and sufficient condition for the addition of each such edge (u, v). Before we present it, we need the following definition for ease of notation.

Definition 2. *Given a chordal graph $G = (V, E)$ and any pair of vertices u and v in G such that $(u, v) \notin E$, $R(G, u, v) = \{(u, x) \mid x$ belongs to a minimal uv-separator of $G\}$. We will call $R(G, u, v)$* the set of required edges *for (u, v) incident to u.*

Theorem 5. *Let $G = (V, E)$ be a chordal graph, let u and v be any two non-adjacent vertices of G, and assume that one wants to add edge (u, v) to G. $R(G, u, v)$ is an inclusion minimal set of edges that must be added to G along with edge (u, v) in order to obtain a chordal graph H.*

Proof. Assume that edge set $R(G, u, v) \cup \{(u, v)\}$ is added to G. We will show that H thus obtained is chordal. Observe first that (u, v) is mono saturating in H since every possible uv-separator of H is contained in $N_H(u)$, and thus does not appear on any chordless cycle of length more than 3, as we have seen in the proof of Theorem 4. Assume on the contrary that H is not chordal, and let C be a chordless cycle of length at least 4 in H. Then C must contain at least one newly added edge $(u, x) \in R(G, u, v)$. Let $C = u - y_1 - y_2 - ... - y_k - x - u$ with $k \geq 2$. Since edge (u, x) was added, x belongs to a minimal uv-separator S of G. Let C_1 and C_2 be two connected components of $G(V \setminus S)$ such that $N_G(C_1) = N_G(C_2) = S$. Assume without loss of generality that $u \in C_1$ and $v \in C_2$. Then every vertex of C belongs to $S \cup C_1$ since S is a clique in both G and H, and C is chordless. Thus in G, there is a chordless path p_1 containing $y_2 - ... - y_k - x$ between u and x, where all vertices of p_1 belong to $C_1 \cup S$. In addition there is also a chordless path p_2 between x and v (might be a single edge) passing only through vertices belonging to C_2. As a consequence, y_2 must belong to some minimal ux-separator of G, and thus also to some minimal uv-separator of G, since $p_1 - p_2$ is a chordless path between u and v in G. But then (u, y_2) belongs to $R(G, u, v)$ and has been added to H contradicting our assumption that $C = u - y_1 - y_2 - ... - y_k - x - u$ is a chordless cycle of H.

Now we will show that the set $R(G, u, v)$ is inclusion minimal. Assume on the contrary that (u, v) and a proper subset of $R(G, u, v)$ are added to G, and that the resulting graph H is chordal. Thus there is a vertex x belonging to a minimal uv-separator S of G such that (u, x) does not belong to H. Let C_1 be the

connected component of $G(V \setminus S)$ that contains u and C_2 the one that contains v. In G there must be a chordless path p_1 between u and x with all intermediate vertices belonging to C_1, and p_2 between x and v with all intermediate vertices belonging to C_2. Let q_1 be a vertex of p_1 closest to x and adjacent to u, and let q_2 be an analogous vertex of p_2 (q_2 might be v), such that $u - p_1 - x - p_2 - v = u - p_{11} - q_1 - p_{12} - x - p_{21} - q_2 - p_{22} - v$. Then $u - q_1 - p_{12} - x - p_{21} - q_2 - u$ is a chordless cycle in H , giving us the desired contradiction.

Corollary 2. *Let $G = (V, E)$ be a chordal graph, and let u and v be any pair of non adjacent vertices in G. Then $H = (V,\ E \cup \{(u, v)\} \cup R(G, u, v))$ is a minimal triangulation of $(V,\ E \cup \{(u, v)\})$.*

4 A Vertex Incremental Algorithm for Simultaneous Maximal Subtriangulation and Minimal Triangulation

In this section we apply our results of Section 3 to the problem of computing a maximal chordal subgraph $H = (V, D)$ and a minimal triangulation $M = (V, F)$ of an arbitrary graph $G = (V, E)$, where $D \subseteq E \subseteq F$.

Our algorithm is based on the following vertex incremental principle. Start with an empty subset U of V, increase U with a new vertex u of G at each step, and do computations according to Theorem 5 to obtain a maximal chordal subgraph H of $G(U)$ or a minimal triangulation M of $G(U)$ on vertex set U at the end of each step. In the case of a maximal subtriangulation, we will allow adding only edges that belong to E between u and the vertices of H, whereas in the case of a minimal triangulation, the required edges between u and H will also be added. For this incremental approach, we first need the following two lemmas.

Lemma 1. *Given $G = (V, E)$, let $H = (U, D)$ be a maximal chordal subgraph of $G(U) = (U, E')$, where $U \subset V$ and $D \subseteq E' \subseteq E$. No edge belonging to $E' \setminus D$ can be contained in a maximal chordal subgraph of G that also contains H.*

Proof. Let (u, v) be any edge of $E' \setminus D$. Thus u and v both belong to the chordal subgraph H. Let $H' = (V, D')$ be a maximal chordal subgraph of G with $D \subset D'$, and assume on the contrary that (u, v) belongs to D'. Since induced subgraphs of chordal graphs are also chordal, $H'(U)$ is chordal and contains edge (u, v). But this contradicts the assumption that H is a maximal chordal subgraph of $G(U)$, since $H'(U)$ is a chordal subgraph of $G(U)$ that contains H as a proper subgraph.

Lemma 2. *Given $G = (V, E)$, let $M = (U, F)$ be a minimal triangulation of $G(U)$ with $U \subset V$. Then any minimal triangulation of $(V, E \cup F)$ obtained by introducing only edges with at least one endpoint in $V \setminus U$ is a minimal triangulation of G.*

Proof. Let $M' = (V, F')$ be a minimal triangulation of $(V, E \cup F)$ obtained by introducing only edges with at least one endpoint in $V \setminus U$. M' exists by Theorem 5, and M' is certainly a triangulation of G since it is chordal and contains all edges of G. Assume on the contrary that M' is not a minimal triangulation of G. Thus there is at least one edge in $F' \setminus E$ that can be removed. If this edge belongs to $F' \setminus (E \cup F)$, then this contradicts our assumption that M' is a minimal triangulation of $(V, E \cup F)$. Thus an edge (u, v) belonging to $F \setminus E$ can be removed from M' without destroying its chordality. However, since M is a minimal triangulation of $G(U)$, removing (u, v) creates a chordless cycle C of length at least 4 in M. Since no edge of $F' \setminus F$ have both its endpoints in U, $F' \setminus F$ does not contain a chord of C, and consequently the vertices belonging to C will induce a chordless cycle in M' if (u, v) is removed, giving the desired contradiction.

With the data structure proposed in the next section, computing and adding set $R(H, u, v)$ can be done in $O(n)$ time for each examined edge (u, v). Observe that every edge needs to be examined at most once, giving a total time complexity of $= O(nm)$. We are now ready to present our algorithm, and here we give the maximal chordal subgraph version.

Algorithm Incremental Maximal Subtriangulation **(IMS)**
Input: $G = (V, E)$.
Output: A maximal chordal subgraph $H = (V, D)$ of G.
Pick a vertex s of G; $U = \{s\}$; $D = \emptyset$;
for $i = 2$ to n **do**
  Pick a vertex $u \in N_G(U)$; $U = U \cup \{u\}$; $N = N_G(u) \cap U$;
  while N is not empty **do**
    Pick a vertex $v \in N$; $N = N \setminus \{v\}$;
    $X = \{x \mid x$ belongs to a minimal uv-separator of $H = (U, D)\}$;
    $R = \{(u, x) \mid x \in X\}$;
    if $R \subseteq E$ **then**
      $D = D \cup \{(u, v)\} \cup R$; $N = N \setminus X$;
  $H = (U, D)$;

Let us call **IMT** (Incremental Minimal Triangulation) the algorithm that results from removing line "**if** $R \subseteq E$ **then**" of Algorithm **IMS**. Thus in **IMT**, edge set $\{(u, v)\} \cup R$ is always added to the transitory graph for every examined edge (u, v). It can be proved by straight forward induction using Theorem 5 and Lemmas 1 and 2 that Algorithm **IMS** computes a maximal chordal subgraph and Algorithm **IMT** computes a minimal triangulation of the input graph. In Example 1, executions of both of these algorithms are shown on the same input graph. Figure 1 (a) shows **IMS** and (b) shows **IMT**.

Example 1. Consider Figure 1. The vertices of the input graph are processed in the order shown by the numbers on the vertices. At step 1, only vertex 1 is added to H. At step 2, vertex 2 and edge $(2, 1)$ are added, and similarly at steps 3 and 4, vertex 3 and edge $(3, 2)$, and vertex 4 and edge $(4, 1)$ are added, respectively. The first column of the figure shows graph H with thick lines on the

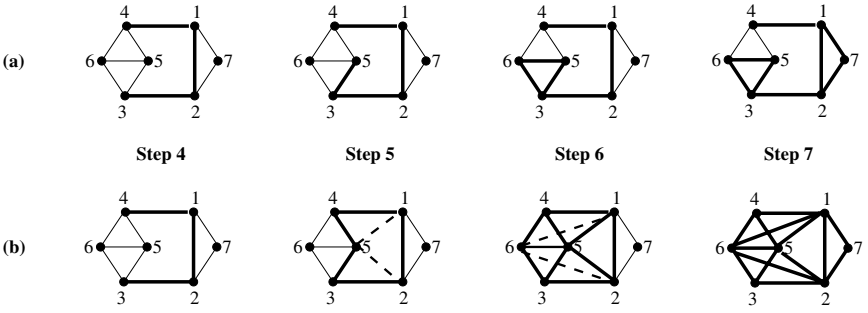

Fig. 1. The figure shows graph H in thick lines after steps 4, 5, 6, and 7 of (a) Algorithm **IMS** when computing a maximal chordal subgraph and (b) Algorithm **IMT** when computing a minimal triangulation.

input graph after these 4 steps. Graph H so far is the same for both a maximal chordal subgraph (a), and a minimal triangulation (b). We will explain the rest of the executions in more detail.

(a) At step 5, $N = \{3, 4\}$, and edge $(5, 3)$ is examined first. In this case, set X is empty, and edge $(5, 3)$ is thus added. For the addition of edge $(5, 4)$, $X = \{1, 2, 3\}$, and since required edges $(5, 1)$ and $(5, 2)$ are not present in G, edge $(5, 4)$ is not added. At step 6, $N = \{3, 4, 5\}$, and edge $(6, 3)$ is examined first and added since X is empty. For the addition of edge $(6, 4)$, $X = \{1, 2, 3\}$, and since required edges $(6, 1)$ and $(6, 2)$ are not present in G, edge $(6, 4)$ is not added. For the addition of edge $(6, 5)$, $X = \{3\}$, and $(6, 5)$ is added since edge $(6, 3)$ is present in G and in H.

(b) At step 5, edge $(5, 3)$ is added as in (a), and in addition, edge $(5, 4)$ is added along with the required edges $(5, 1)$ and $(5, 2)$. At step 6, edge $(6, 3)$ is added as in (a). For the addition of edge $(6, 4)$, $X = \{1, 2, 3, 5\}$ since the minimal $6, 4$-separators are $\{1, 5\}, \{2, 5\}$, and $\{3\}$. Thus edge $(6, 4)$ and required edges $(6, 1)$, $(6, 2)$, and $(6, 5)$ are added to H.

Step 7 adds edges $(7, 1)$ and $(7, 2)$ in both (a) and (b) without requiring any additional edges in either case.

5 Data Structure and Time Complexity

The input graph G is represented by an adjacency list, and we use a clique tree T of H as an additional data structure to store and work on H. In order to achieve the total $O(nm)$ time bound, for each edge (u, v) of G to be examined we have to do the following two operations in $O(n)$ time: 1. Compute the union X of all minimal uv-separators in H, which gives the required edge set $R(H, u, v)$. 2. If $R(H, u, v) \cup \{(u, v)\}$ is to be added to H, update T to reflect this modification of H.

The main idea is to use a path P_{uv} of the clique tree T between a clique (tree node) C_u that contains u and a clique C_v that contains v, and compute

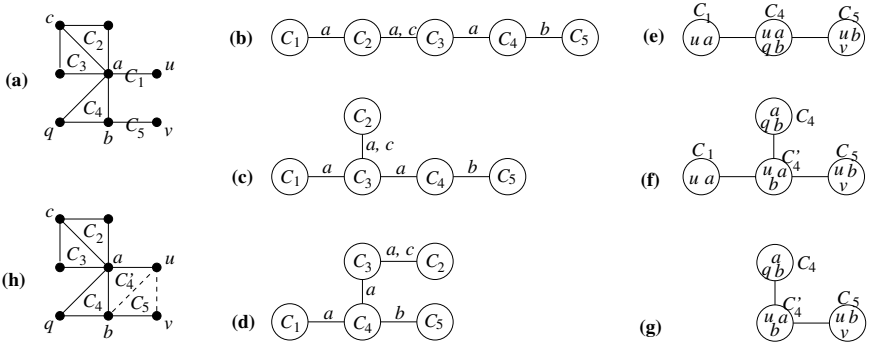

Fig. 2. A chordal graph H is given in (a), and (b) shows a clique tree of H, where $C_u = C_1$, $C_v = C_5$, and P_{uv} is the whole clique tree. After steps (c) and (d), path P_{uv} between C_1 and C_5 is in the desired form, and only this portion of the tree is shown after step (d). In step (e), u is placed in every clique on P_{uv}, and in step (f) C_4 is separated from the path since edge (u, q) is not intended. C_1 is removed in (g) since it becomes non maximal. The new corresponding graph H of which the modified tree is a clique tree is shown in (h).

the union of edges on this path that correspond to minimal uv-separators. For this operation, C_u and C_v are chosen so that no maximal clique between C_u and C_v on P_{uv} contains u or v, as illustrated in Figure 2 (a) and (b). In addition, T is modified so that path P_{uv} between C_u and C_v contains only distinct minimal uv-separators, as shown in Figure 2 (c) and (d). Now the union of the edges of P_{uv} give the desired vertex set X.

Unfortunately, the sum of the sizes of the edges on P_{uv} can be larger than $O(n)$. Thus if the tree nodes and tree edges of T are implemented in the traditional way as vertex lists containing vertices of each tree node and edge, then Operation 1 described above cannot be done in $O(n)$ time. For this reason, we present a new implementation of clique trees: Every edge (C_1, C_2) of T is implemented as two lists that we will call *difference lists*. One list contains vertices belonging to $C_1 \setminus C_2$. This list has two names; it is called both $add(C_2, C_1)$ and $remove(C_1, C_2)$. The other list contains vertices belonging to $C_2 \setminus C_1$. This list is called $add(C_1, C_2)$ and also $remove(C_2, C_1)$. Now, if every clique C of T contains pointers to its add and remove lists, then C actually does not need to store a list of vertices that it contains. To see how to compute X, let $P_{uv} = C_1 - C_2 - \ldots - C_k$; then $X = \cup_{i=2}^{k-1}(add(C_{i-1}, C_i) \setminus remove(C_i, C_{i+1}))$. Every vertex of G can appear in at most one add list and at most one remove list on this path due to Theorem 3, and thus computing X can be done in $O(n)$ time as described using a characteristic vector to store X. Further details of Operation 1, and details of Operation 2 are omitted in this extended abstract due to limited space. We refer the reader to Figure 2 (e) - (h).

6 Concluding Remarks

In this paper, we contribute new theoretical results on chordality as well as an efficient handling of the corresponding data structures. Not only do we have a new $O(nm)$ time dynamic algorithm for minimal triangulation of a graph G, but we are able to compute at the same time a maximal chordal subgraph, thus "minimally sandwiching" the graph between two chordal graphs: $H_1 \subseteq G \subseteq H_2$. This special feature of our algorithm enables the user, at no extra cost, to choose at each vertex addition step whether he wants to *add* or *delete* edges, or even to do so at each edge addition step.

When one wants to add to a chordal graph an edge between $\{u, v\}$ which is not a 2-pair, there is a succession of 2-pair edges incident to u that can be added first, making $\{u, v\}$ a 2-pair of the new graph thus obtained. Our main theorem precisely describes this set of edges as our set of required edges. It was not known earlier which edges needed to be added in order to ensure that $\{u, v\}$ becomes a 2-pair, and even less how to compute them efficiently.

References

1. S. Arikati and P. Rangan. An efficient algorithm for finding a two-pair, and its applications. *Disc. Appl. Math. Comb. Oper. Res.*, 31:71–74, 1991.
2. E. Balas. A fast algorithm for finding an edge-maximal subgraph with a TR-formative coloring. *Disc. Appl. Math.*, 15:123–134, 1986.
3. A. Berry. A wide-range efficient algorithm for minimal triangulation. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999.
4. A. Berry, J. Blair, and P. Heggernes. Maximum cardinality search for computing minimal triangulations. In L. Kucera, editor, *Graph Theoretical Concepts in Computer Science - WG 2002, LNCS 2573*, pages 1–12. Springer Verlag, 2002.
5. A. Berry, J-P. Bordat, P. Heggernes, G. Simonet, and Y. Villanger. A wide-range algorithm for minimal triangulation from an arbitrary ordering. Technical Report Reports in Informatics 243, University of Bergen, Norway, 2003.
6. A. Berry, A. Sigayret, and C. Sinoquet. Maximal sub-triangulation as improving phylogenetic data. Technical Report RR-02-02, LIMOS, Clermont-Ferrand, France, 2002.
7. J. R. S. Blair, P. Heggernes, and J. A. Telle. A practical algorithm for making filled graphs minimal. *Theoretical Computer Science*, 250:125–141, 2001.
8. J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. In J. A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computations*, pages 1–30. Springer Verlag, 1993. IMA Volumes in Mathematics and its Applications, Vol. 56.
9. P. Buneman. A characterization of rigid circuit graphs. *Discrete Math.*, 9:205–212, 1974.
10. E. Dahlhaus. Minimal elimination ordering inside a given chordal graph. In R. H. Möhring, editor, *Graph Theoretical Concepts in Computer Science - WG '97, LNCS 1335*, pages 132–143. Springer Verlag, 1997.
11. P. M. Dearing, D. R. Shier, and D. D. Warner. Maximal chordal subgraphs. *Disc. Appl. Math.*, 20:181–190, 1988.
12. G. A. Dirac. On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg*, 25:71–76, 1961.

13. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combin. Theory Ser. B*, 16:47–56, 1974.
14. R. Hayward, C. Hoàng, and F. Maffray. Optimizing weakly triangulated graphs. *Graphs and Combinatorics*, 5:339–349, 1989.
15. C-W. Ho and R. C. T. Lee. Counting clique trees and computing perfect elimination schemes in parallel. *Inform. Process. Lett.*, 31:61–68, 1989.
16. L. Ibarra. Fully dynamic algorithms for chordal graphs. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999.
17. A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Disc. Appl. Math.*, 113:109–128, 2001.
18. D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266–283, 1976.
19. J. Spinrad and R. Sritharan. Algorithms for weakly triangulated graphs. *Disc. Appl. Math.*, 59:181–191, 1995.
20. J. Walter. *Representations of rigid cycle graphs.* PhD thesis, Wayne State University, USA, 1972.
21. J. Xue. Edge-maximal triangulated subgraphs and heuristics for the maximum clique problem. *Networks*, 24:109–120, 1994.
22. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.