Chordal graphs and their clique graphs

Philippe Galinier, Michel Habib and Christophe Paul

LIRMM
UMR 9928 UNIVERSITE MONTPELLIER II/CNRS
161, Rue Ada
34392 Montpellier cedex 5 France
email: paul@lirmm.fr

Abstract. In the first part of this paper, a new structure for chordal graph is introduced, namely the clique graph. This structure is shown to be optimal with regard to the set of clique trees. The greedy aspect of the recognition algorithms of chordal graphs is studied. A new greedy algorithm that generalizes both Maximal cardinality Search (MCS) and Lexicographic Breadth first search is presented. The trace of an execution of MCS is defined and used in two linear time and space algorithms: one builds a clique tree of a chordal graph and the other is a simple recognition procedure of chordal graphs.

Introduction

Since chordal graphs have no chordless cycle of length more than 3, they can be considered as a generalization of trees. In [9], chordal graphs have been considered as the intersection graphs of subtrees of a tree. Chordal graphs are often represented by a clique tree (see [9, 19]). This is a structure which translates most of the information contained in a chordal graph. The structure of clique tree does not only appeared in the graph theory litterature, but in the context of the acyclic database schemes [6, 1] and in the context of sparse matrix computations too [3, 12, 14]. This is probably why some results have been rediscovered independently several times. Some results of this paper are contained in [3], but we present here a unifying graph theoretical point of view.

Chordal graphs can also be characterized using perfect elimination orderings (PEO) [16]. A vertex is simplicial if and only if its neighbourhood is a complete subgraph. An elimination ordering $x_1, x_2, ..., x_n$ is perfect if and only if each x_i is simplicial in the subgraph induced by $x_i, ..., x_n$.

Several greedy recognition algorithms of chordal graphs are known. The most famous are the Lexicographic Breadth First Search (BFS for short) [17] and the Maximum Cardinality Search (MCS for short) [20]. Chordal graphs can be represented with clique-trees (see for example [19]). The linear recognition of chordal graphs involves two distinct phases: the execution of MCS or BFS in order to compute an elimination ordering and a checking procedure to decide whether this elimination ordering is perfect (PEO).

In the first part of this paper, a new structure namely the clique graph is introduced. Some graph properties of this structure are studied with regard to clique trees. And the clique graph is justified as being the optimal structure containing all clique trees of a chordal graph. In the second section, an explanation of the greedy aspect of the recognition algorithms, MCS and BFS, is given. The strong properties of the clique graph allow a simple algorithm which computes maximum weighted spanning tree. It is shown that such an algorithm generalizes both MCS or BFS. In fact, in [3] MCS has yet been compared to Prim's algortihm [15]. This correspondence gives a better knowledge of the greediness of chordal graph recognitions and implies new proof for these algorithms. In the last part of this paper, we present two linear algorithms: one for building a clique-tree, and the other for the recognition of PEO. We define the trace of an elimination ordering as the mark level of vertices when they are numbered by MCS. Both algorithms are based on the study of the increasing sequences of the trace induced by the elimination ordering . This study produces a new and simplier understanding of the recognition of chordal graphs. These two algorithms can also be adapted for BFS.

1 The clique graph of a chordal graph

One of the most widely used representations of chordal graphs is the clique tree defined above (see [9]). In this section, we will introduce and study a new structure called the *clique graph* of a chordal graph. We will show the ties between clique graphs and the clique trees. In [19], the clique tree is studied with regard to the clique intersection graph, which can be seen as the cliques hypergraph [2]. The clique graph defined here is a subgraph of the clique intersection graph. We will prove that a clique graph can be seen as the minimal graph containing all clique trees. All graphs considered here are supposed to be connected, if not each connected component has to be considered separately.

Definition 1. Given an undirected graph $G = (V, E)$, and two non-adjacent vertices a and b, a subset $S \subset V$ is an a, b-separator if the removal of S separates a and b in distinct connected components. If no proper subset of S is an a, b-separator then S is a minimal a, b-separator. A (minimal) separator is a set of vertices S for which there exist non adjacent vertices a and b such that S is a (minimal) a, b-separator.

It is well known [8], that the minimal separators of a chordal graphs are complete subgraphs. Here is the definition of the clique graph of a chordal graph.

Definition 2. Let $G = (V, E)$ be a chordal graph. The clique-graph of G, denoted by $C(G) = (V_c, E_c, \mu)$, with $\mu : E_c \to N$, is defined as follows :

1. The vertex set V_c, is the set of maximal cliques of G;
2. The edge (C_1, C_2) belongs to E_c if and only if the intersection $C_1 \cap C_2$ is a minimal a, b-separator for each $a \in (C_1 \setminus C_2)$ and each $b \in (C_2 \setminus C_1)$;
3. The edges of $(C_i, C_j) \in V_c$ are weighted by the cardinality of the corresponding minimal separator $S_{ij} : \mu(C_i, C_j) = |S_{ij}|$.

Let C_i and C_j be two maximal cliques of a chordal graph. Hereafter, we will note $S_{ij} = C_i \cap C_j$ if and only if S_{ij} is a minimal a,b-separator for each $a \in (C_i \backslash C_j)$ and each $b \in (C_j \backslash C_i)$. Let us now prove several structure properties of the clique graph.

Triangle Lemma Let (C_1, C_2, C_3) be a 3-cycle in $C(G)$ and let S_{12}, S_{13}, S_{23} be the associated minimal separators of G. Then two of these three minimal separators are equal and included in the third.

Proof: Assume that two minimal separators among S_{12}, S_{13}, S_{23} are incomparable for the inclusion order. Let S_{12} and S_{13} be these minimal separators. Then there exist two vertices x and y such that $x \in (S_{12} \setminus S_{13})$ and $y \in (S_{13} \setminus S_{12})$. Since C_1, C_2, C_3 are distinct maximal cliques, $C_2 \setminus C_3$ and $C_3 \setminus C_2$ are not empty. The vertices x and y do not belong to $C_2 \cap C_3$. For each $a \in (C_2 \setminus C_3)$ and each $b \in (C_3 \setminus C_2)$, the path a, x, y, b exists and is not cut by $C_2 \cap C_3$. A contradiction, therefore $C_2 \cap C_3$ is not a a,b-separator and the edge between C_2 and C_3 does not exist.

Therefore if there exists a 3-cycle in $C(G)$, then the three minimal separators on the edges can be linearly ordered by inclusion. Without loss of generality, assume that $S_{12} \subset S_{13} \subseteq S_{23}$. Therefore $S_{13} \subset C_1$ and $S_{13} \subset C_2$. And so $S_{13} \subset (C_1 \cap C_2)$. This leads to a contradiction : $S_{13} \subset S_{12}$. We have proved that $S_{12} = S_{13} \subseteq S_{23}$. □

Note that the converse is false. Let C_1, C_2, C_3 be three maximal cliques such that $(C_1, C_2) \in E_c$ and $(C_1, C_3) \in E_c$, then $S_{12} = S_{13}$ does not imply that the edge $(C_2, C_3) \in E_c$. But the following property stands.

Lemma 3. *Let $C(G)$ be the clique graph of the chordal graph G. Let C_1, C_2, C_3 be three maximal cliques such that $(C_1, C_2) \in E_c$ and $(C_1, C_3) \in E_c$, then $S_{12} \subset S_{13} \Rightarrow (C_2, C_3) \in E_c$.*

Proof: By the triangle lemma, $S_{12} \subset S_{13}$ (strict inclusion) implies that $C_2 \cap C_3 = S_{12}$, otherwise there is no edge between C_1 and C_2. By definition S_{12} is a minimal separator for all $a \in C_2 \setminus C_1$ and $b \in C_1 \setminus C_2$. Since $S_{12} \subset S_{13}$, every vertex $c \in C_3 \setminus C_2$ is in the same connected component of $G[V - S_{12}]$ than b. And so $C_2 \cap C_3$ is a minimal separator for every vertices like a and c. □

The clique graph of a chordal graph is not always chordal. Let us now examine the structure of clique trees in full details. After recalling the definition, we will show that a kind of chordality property of the clique tree stands.

Definition 4. Let $G = (V, E)$ be a chordal graph. A clique tree of G is a tree $T_C = (C, F)$ such that C is the set of maximal cliques of G and for each vertex $x \in E$, the set of maximal cliques containing x induces a subtree of T_C.

Lemma 5. *Let T be a clique tree of the chordal graph G and let C_1 and C_2 be two adjacent maximal cliques. Then $C_1 \cap C_2$ is a minimal separator for all $a \in C_1 \setminus C_2$ and $b \in C_2 \setminus C_1$.*

Proof: If $C_1 \cap C_2$ is not an a,b-separator, then there exists a path between a and b which avoids $C_1 \cap C_2$. Every edge of this path belongs to a maximal clique. Therefore a and b are contained in some of these maximal cliques. But T is a tree, hence the set of maximal cliques containing a or b can not induce a subtree of T. So $C_1 \cap C_2$ is an a,b-separator and is necessarily minimal otherwise C_1 or C_2 should not be a clique. □

Weak Triangulation Lemma Let $P_{ik} = [C_1, ..., C_k]$, $k \geq 4$, be a path in a clique tree T of a chordal graph G. If (C_1, C_k) is an edge of $C(G)$, then either (C_2, C_k) or (C_1, C_{k-1}) is an edge of $C(G)$.
Proof:

For each $1 \leq i \leq k-1$, $S_{i,i+1}$ is a a,b-minimal separator for each $a \in C_i \setminus C_{i+1}$ and each $b \in C_{i+1} \setminus C_i$. Then by lemma 5 each $S_{i,i+1}$ is a x,y-separator for $x \in C_1 \setminus C_2$ and $y \in C_k \setminus C_{k-1}$, but not necessarily minimal. Since $(C_1, C_k) \in V_c$, S_{1k} is a minimal x,y-separator. Hence S_{1k} must be included in all $S_{i,i+1}$. The triangle lemma proves the existence of a chord in the cycle $C_1, ..., C_k$ in $C(G)$. And so the triangle lemma applied iteratively, proves that either (C_2, C_k) or (C_1, C_{k-1}) is an edge of $C(G)$. □

In other words, the above lemma shows that any path of a clique tree induces a chordal subgraph of $C(G)$. Next theorem shows ties between clique trees and maximum weighted spanning trees of $C(G)$.

Theorem 6. *Let $G = (V, E)$ be a chordal graph and $C(G)$ its clique graph. Let $T = (V_c, F)$ be a spanning tree of $C(G)$. Then T is a clique tree of G if and only if T is a maximum weighted spanning tree of $C(G)$.*

Proof: Since the set of nodes of T and $C(G)$ are the same, we just need to prove that the set of nodes containing a vertex $x \in E$ induces a subtree of T if and only if T is a maximum weighted spanning tree of $C(G)$.

$\Rightarrow$ Assume that T is not a maximum weighted spanning-tree of $C(G)$. Then there exists a pair of maximal cliques, C_i and C_j, adjacent in $C(G)$ and not in T such that S_{ij} strictly contains a minimal separator of the unique path P_{ij} between C_i and C_j in T. Let S_{kl} be such a minimal separator on P_{ij}. Let $x \in (S_{ij} \setminus S_{kl})$. So x does not belong to at least one of the two nodes C_k and C_l. Hence, the set of nodes containing the vertex x is not a subtree of T and so T is not a clique-tree of G.

$\Leftarrow$ Assume that T is a maximum weighted spanning-tree of $C(G)$. Let C_i and C_j be two non-adjacent maximal cliques of T containing the vertex $x \in V$. Assume that x does not belong to any maximal cliques on the unique path in T between C_i and C_j, P_{ij}.
Consider the subgraph G' of G induced by the maximal cliques of P_{ij}. The subgraph G' is a chordal graph. In [16], it is proved that each vertex of a chordal graph is either a simplicial vertex, or belongs to a minimal separator. Simplicial vertices belong to only one maximal clique. Therefore there exists a minimal separator S of G' such that $x \in S$. Since no maximal clique apart from C_i and C_j contains x, S must be equal to $C_i \cap C_j$. Hence the edge

$(C_i, C_j) \in E_c$, and the maximal cliques of P_{ij} induces a cycle. Let $y \in C_i \setminus S$ and $z \in C_j \setminus S$ be two vertices, then S is a minimal z, y-separator. If no minimal separator on P_{ij} is strictly included in S, we can easily extract a path in G' between y and z which is not cut by S. This leads to a contradiction. Therefore there exists S' on P_{ij} such that $S' \subset S$. Hence T is not a maximum spanning tree. So x belongs to any maximal clique on P_{ij}, and T is a clique tree.

□

The next theorem yields a kind of optimality for a clique graph.

Theorem 7. *Let $C(G)$ be the clique graph of the chordal graph G. Then each edge of $C(G)$ belongs to at least one clique tree.*

Proof: Let T be a clique tree of G. And let C_i and C_k two adjacent maximal cliques in $C(G)$ but not in T. Let P_{ik} be the path between C_i and C_k in T. By the weak triangulation lemma, either the edge (C_1, C_{k-1}) or (C_2, C_k) exists. Assume that (C_1, C_{k-1}) exists. Then C_1, C_k, C_{k-1} is a 3-cycle, and so the triangle lemma can be applied. Since T is a maximum spanning tree, (C_1, C_k) is not the biggest edge of the 3-cycle. So $S_{1,k} = S_{1,k-1}$. Applying the weak triangulation lemma iteratively, we can find an edge E in P_{ij} with the same label than (C_1, C_k). Therefore changing E by (C_1, C_k) in T yields to a new clique tree. We have proved that every edge of $C(G)$ can belong to a clique tree. □

The previous theorem proves that the edges belonging to the clique intersection graph and not to the clique graph, never belong to a clique tree. Since every edges of a clique tree are minimal separator, clique graphs are the minimal structures containing all clique trees.

2 Greedy aspects of recognition algorithms

Let $G = (V, E)$ be a graph with n vertices. When an elimination ordering is computed by BFS or MCS, another procedure must verify if it is a perfect elimination ordering in order to prove that G is triangulated. BFS or MCS computes the elimination ordering in the reverse order. In this section, we will give an explanation of the greedy aspect of the two linear recognition algorithms of chordal graphs: MCS and BFS. The main result proves that both algorithms compute a maximum spanning-tree of the clique graph.

Let us examine how MCS and BFS visit a chordal graph. We just give the proof for MCS, but this proof can be easily transformed for BFS. When MCS chooses a new vertex x, then the mark level of this vertex, noted $mark(x)$, is maximum over all unnumbered vertices. The set of vertices who has marked x will be denoted by $M(x)$.

Lemma 8. *Let $G = (V, E)$ be a chordal graph. In a execution of MCS or BFS on G, maximal cliques of G are visited consecutively.*

Proof: Let α the PEO computed by MCS. Let $N = \{x_n, ..., x_i\}$ be the set of numbered vertices at some step. Then x_i is simplicial in $G[x_n, ..., x_i]$, and so x_i belongs to a unique maximal clique (see [16]). Let us prove that x_{i-1} belongs to a new maximal clique if and only if $mark(x_{i-1}) \leq mark(x_i)$.

- $\Leftarrow$ Assume that x_{i-1} and x_i belong to the same maximal clique. Since x_i is simplicial in $G[x_n, ..., x_i]$, all the vertices of $M(x_i)$ belong to this maximal clique. Hence $mark(x_{i-1}) = mark(x_i) + 1$.
- $\Rightarrow$ Assume that x_{i-1} belongs to a new maximal clique. Since x_i was a vertex with the biggest level mark over all unnumbered vertices when it was choosen, $mark(x_i) \geq |M(x_{i-1}) \setminus \{x_i\}|$. But there exist at least one vertex of the maximal clique containing x_i in $G[x_n, ..., x_i]$ which does not mark x_{i-1}, otherwise x_{i-1} does not belong to a new maximal clique. Therefore $mark(x_{i-1}) \leq mark(x_i)$.

If we define the trace of α as the sequence of mark levels of the vertices when they are numbered, each maximal clique is represented by an increasing sequence. We can conclude that the maximal cliques are visited consecutively. A similar argument holds for BFS. $\square$

Now the remaining question is, in which ordering the maximal cliques of the input chordal graph are visited by both algorithms? The edges of the clique graph of a chordal graph $G = (V, E)$ represent all possible transitions from a maximal clique to another one.

In fact a clique graph is a very particular weighted graph. Therefore there are many ways for computing maximum weighted spanning trees in $C(G)$. The following algorithm is one of them. We will prove that BFS and MCS can be seen as special case of this algorithm. In order to find such a general framework, in the following algorithm we assume that the edges of $C(G)$ are labelled with the minimal separators (until now it was only required that the edges of $C(G)$ to be weighted by the cardinality of the minimal separator).

Algorithm 1: Maximum weighted spanning-tree of clique graph

```
**Data**: A clique graph $C(G)$
**Result**: A maximum spanning-tree of $C(G)$
Choose a maximal clique $C_1$
**for** $i = 2$ *to* $n$ **do**
    choose a maximally (under inclusion) labelled edge adjacent to $C_1, ..., C_{i-1}$
    to connect the new clique $C_i$
```

Theorem 9. *Let G be a chordal graph, then algorithm 1 computes a maximum weighted spanning tree (i.e. a clique tree) of the clique graph $C(G)$.*

Proof: Let T_i be the tree built by algorithm 1 at step i. We now prove by induction the following invariant: *T_i can be completed as a maximum spanning tree of $C(G)$.*

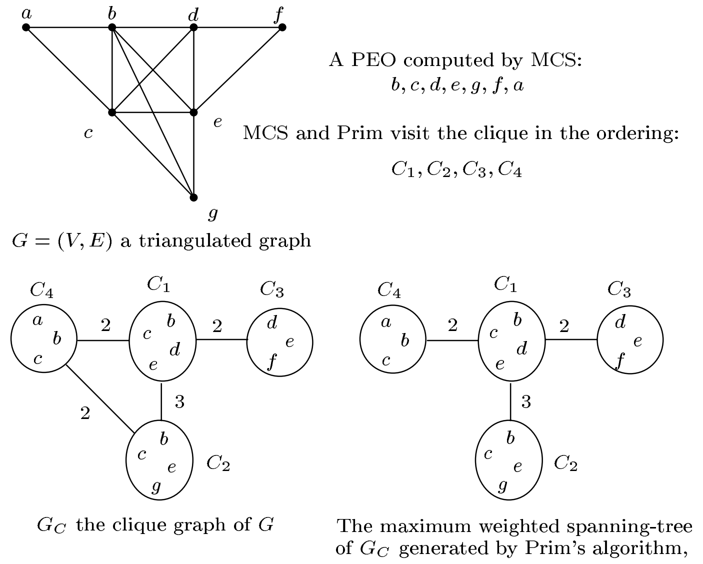
Clearly the property holds for T_1. Let us suppose by induction, it holds for T_{i-1} and that (C_j, C_i), with $j < i$ is the edge chosen by the algorithm at step i. Therefore it exists a maximum spanning tree T containing T_{i-1}.

If (C_j, C_i) belongs to T, we have finished. Else it exists a unique path $\mu = [C_j = D_1, \ldots, D_k = C_i]$ from C_j to C_i in T. Let D_h be the last vextex of $C_1, \ldots, C_{i-1}$ belonging to μ. algorithm 1 insures that $A = label[(D_h, D_{h+1})]$ does not contain $B = label[(C_j, C_i)]$. If $\exists b \in B - A$, then $b \in C_j$ and therefore by the definition of clique tree, b must also belong to all cliques in μ, a contradiction.

Therefore $A = B$, and a maximum spanning tree T' can be obtained from T by exchanging the edges (D_h, D_{h+1}) and (C_j, C_i), which finishes the proof.

□

In [3], it is proven that MCS corresponds to Prim's algorithm. It is easy to see that Prim's algorithm is a special case of the algorithm 1. The next corollary proves the same for BFS.



a
b
d
f
c
e
g
A PEO computed by MCS:
b, c, d, e, g, f, a
MCS and Prim visit the clique in the ordering:
C_1, C_2, C_3, C_4
$G = (V, E)$ a triangulated graph
C_4
C_1
C_3
C_2
2
3
G_C the clique graph of G
The maximum weighted spanning-tree of G_C generated by Prim's algorithm,

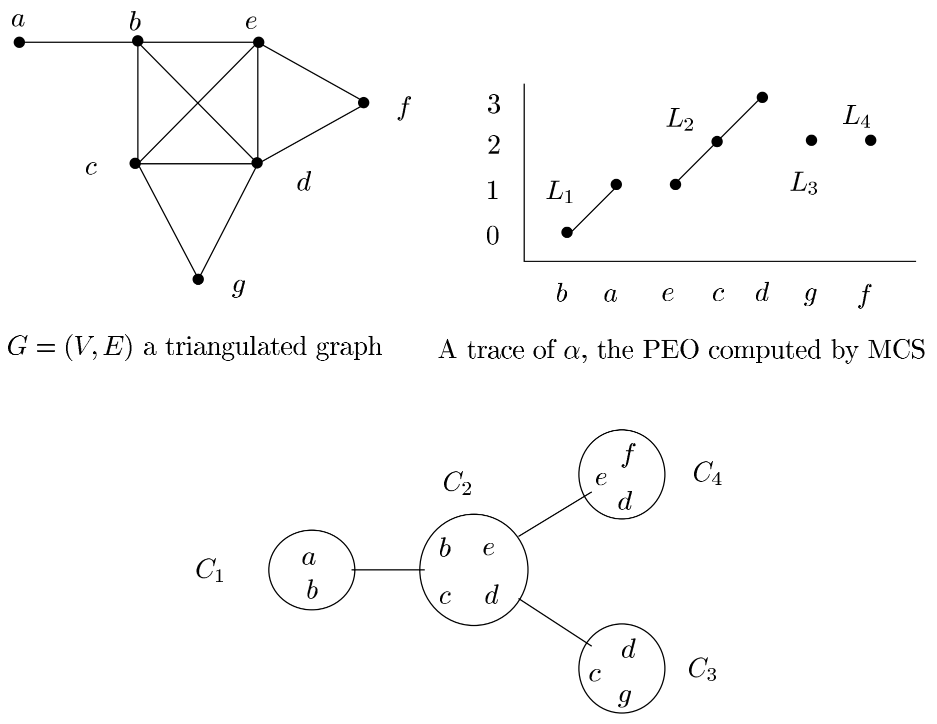
Fig. 1. An execution of MCS on a chordal graph and the corresponding execution of Prim's algorithm on its clique graph

Corollary 10. *MCS and BFS compute maximum spanning trees of $CC(G)$.*

Proof: It suffices to show that MCS and BFS are particular cases of algorithm 1. Let $C_1, ...C_{k-1}$ be the visited cliques and $x_1, ..., x_{i-1}$ be the numbered vertices. When x_i is numbered, a new clique C_k is visited. Since x_i is simplicial in $G[x_1, ..., x_i]$, its neighbourhood is complete and strictly included in a clique C_j, $1 \geq j \geq k-1$. Therefore the labelof all edges connecting C_k to $C_1, ...C_{k-1}$ must be include in the label of (C_i, C_j). In order to connect, C_k the maximum edge is choosen. This is what algorithm 1 does, since a amximum weighted edge has necessarily a maximal label. Similarly for BFS the lexicographic search insures also a maximal labelled edge to be chosen at each step. $\square$

3 Linear algorithms on the clique trees

In this section, we will first present a linear time and space algorithm which computes a clique tree of G will be presented. This algorithm based on MCS. A similar algorithm was presented in [3] We define the trace of an PEO α computed by MCS as the sequence of mark level of the vertices when they are numbered (see figure 2). In the second part, an extended version of the first algorithm for the recognition of PEO is presented. The reader should note that exactly the same work can be done with BFS.



$G = (V, E)$ a triangulated graph

A trace of α, the PEO computed by MCS

The clique-tree associated to α

Fig. 2. An execution of MCS and the associated clique-tree

By the lemma 8, each maximal clique of G corresponds to a strictly increasing sequence of the trace. Let us recall and define some notations. The level mark of a vertex will be denoted by $mark(x)$, and the set of vertices who have marked x is $M(x)$. At each vertex x, we associate $C(x)$, the first maximal clique discovered in α containing x.

Since maximal cliques are visited one after the other, MCS gives an ordering of the maximal cliques: the j-th maximal clique reached by MCS will be noted C_j. At each maximal strictly increasing sequence of marks, L_j in the following (see figure 2), corresponds C_j. Note that the set of L_j is a partition of V. Since α is a PEO, for every x_i, $M(x_i) \subset C(x_i)$. Hence if $x_i \in L_j$, then $C(x_i) = L_j \cup M(x_i) = C_j$.

Let x_i be the first vertex discovered in L_j. Then for every vertex $x_h \in L_j$, $M(x_i) \subseteq M(x_h)$. Henceforth, $mark_j$ will denote the last vertex which has marked all vertices of L_j, i.e $last(x_i)$.

Lemma 11. *Let G be a chordal graph. Let T be a tree whose nodes are the maximal cliques of G, and such that an edge between the maximal cliques C_j and C_k $(k < j)$ exists if and only if $mark_j \in L_k$. Then T is a clique-tree of G.*

Proof: Let x_i be the first vertex discovered in L_j. Let x be a vertex of $M(x_i)$. Let us prove that every maximal clique C on the path between $C(x)$ and C_j in T, contains the vertex x. We have to consider two cases:

1. $C(x)$ and C_j are adjacent (i.e $C(x) = C_k$): trivial.
2. If $C(x)$ and C_j are not adjacent, then $x \neq mark_j$ (i.e. x is not the last vertex which has mark all vertices of C_j). Let us prove that x has marked all vertices of C_k. Since α is a PEO and since $\alpha(x_i) < min(\alpha(x), \alpha(mark_j)$, the edge $(x, mark_j) \in E$. Hence $x \in M(mark_j)$, and so x belongs to C_k.

This proves that the maximal cliques containing x form a subtree of T. Hence T is a clique-tree of G. □

We are now able to present an extended version of MCS which computes on-line the clique-tree associated to the elimination order.

Theorem 12. *The algorithm 2 computes a PEO and its assciated clique-tree if and only if the input graph $G = (V, E)$ is chordal. The complexity of this algorithm is $O(n + m)$ where $n = |V|$ and $m = |E|$.*

Proof: If the alogrithm 2 computes a PEO and a clique-tree, then the input graph is trivially chordal. The algorithm 2 is an extended version of MCS, hence if G is chordal, the computed elimination ordering is perfect. By lemma 11, the step 1 builds a clique-tree if G is chordal.

Let us have a look at the size of a clique tree. First of all, when a vertex is marked by MCS, this mark corresponds to an edge. Hence the size of all mark sets is $O(m)$, the number of edges in G. Since there is at most n increasing sequences, T contains at most n nodes and so $O(n)$ edges.

Let us examine the size of the set of nodes in T. Since the vertices of minimal separators belong to several maximal cliques, the size of the nodes set is bigger than n. Let $M(x)$ be the set of marks of x, where x is the first vertex of an

```
Algorithm 2: Maximum Cardinality Search and Clique-Tree
Data: A graph $G = (V, E)$
Result: If the input graph is chordal: a PEO and an associated clique-tree
$T = (I, F)$ where $I$ is the set of maximal cliques
begin
    each vertex of X is initialized with the empty set
    $previousmark = -1$
    $j = 0$
    for $i = n$ to 1 do
        choose a vertex $x$ not yet numbered such that $|mark(x)|$ is maximum
1       if $mark(x) \leq previousmark$ then
            $j = j + 1$
            create the maximal clique $C_j = M(x) \cup \{x\}$
            create the tie between $C_j$ and $C(last(x))$
        else
2           $C_j = C_j \cup \{x\}$
3       for each $y$ neighbour of $x$ do
            $M(y) = M(y) \cup \{x\}$
            $mark(y) = mark(y) + 1$
            $last(y) = x$
        $previousmark = mark(x)$
        $x$ is numbered by $i$
        $C(x) = j$
end
```

increasing sequence L_j. The node associated to x by the algorithm 2 is $M(x) \cup L_j$. But $M(x)$ corresponds to the edges between vertices of $M(x)$ and x. Since every duplicated vertex belongs to the mark set of the first vertex of an increasing sequence, each of them can be associated an edge. Therefore the sum of the cardinality of the nodes of T is smaller than $n + m$, and the space complexity is $O(n + m)$.

All the operations of step 1, 2 and 3 can be done in constant time. Hence this algorithm has the same time complexity as MCS: $O(n + m)$. □

It is well known that MCS computes a PEO if and only if the graph is chordal. Similarly the results of the previous algorithm stand if and only if the algorithm 2 works on a chordal graph, MCS computes a PEO if and only if the graph is chordal. Hence the natural following question is: how the trace of an elimination ordering computed by MCS can be used for the recognition of chordal graph. The following algorithm checks whether the output of the algorithm 2 is a clique tree or not in linear time and space complexity. Since MCS computes a PEO if and only if the input graph is chordal, then the algorithm 2 builds a clique tree if and only if the elimination ordering is perfect. And so the next algorithm can be seen like a recognition procedure for chordal graphs.

Let T be the result of the algorithm 2. The following two lemmas exhibit the properties of simplicial vertices in T. The figure 3 give an illustration of these properties.

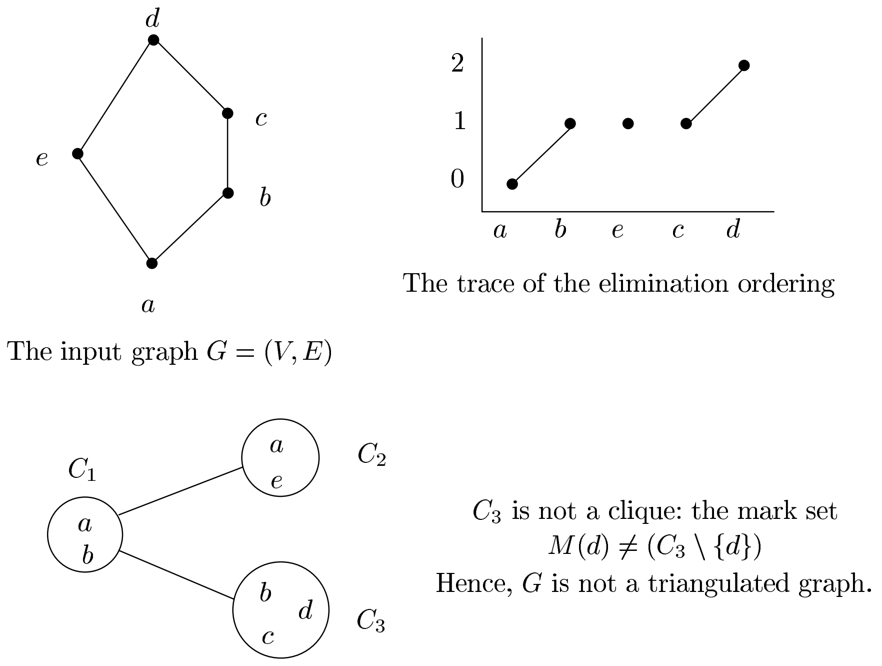

d
c
e
b
a
The input graph $G = (V, E)$
2
1
0
a b e c d
The trace of the elimination ordering
C_1
a
b
a
e
C_2
b
d
c
C_3
C_3 is not a clique: the mark set
$M(d) \neq (C_3 \setminus \{d\})$
Hence, G is not a triangulated graph.

Fig. 3. An execution of the algorithm 3 on a 5-cycle

Lemma 13. *Let x_i be the first vertex of the increasing sequence of marks L_j. Then x_i is simplicial if and only if $M(x_i) \subset C_{father}$.*

Proof: Assume that $M(x_i) \subset C_{father}$ then the neighbourhood of x_i is a clique. If it is not the case, then at least two neighbours of x_i are not in the same clique. Therefore they are not adjacent. Hence its neighbourhood is not a clique. $\square$

Lemma 14. *Let $L_j = \{x_i, ..., x_h\}$ be an increasing sequence of marks such that x_i is simplicial in $G[X_n, ..., x_i]$. Then x_k, with $i < k \leq h$, is simplicial if and only if $M(x_k) = M(x_{k-1} \cup \{x_{k-1}\}$.*

Proof: Let us assume that x_i, the first vertex of the increasing sequence, is simplicial. Then its neighbourhood is a clique. It is clear that x_{i-1} is simplicial if $M(x_{i-1}) = M(x_i) \cup \{x_i\}$. It is the same for each x_k, $i < k \leq h$. If $M(x_k) = M(x_{k-1} \cup \{x_{k-1}\}$, then x_k is simplicial if and only if x_{k-1} is simplicial. Since x_i is simplicial, the property follows. $\square$

These two previous lemmas can lead to an on-line recognition algorithm of chordal graphs. When MCS chooses an unnumbered vertex, one of the two conditions according to the type of this vertex has to be tested. Let us give an idea of such an algorithm. Assume that like in the algorithm 2, the used data structures are sorted lists. Checking whether the first vertex of an increasing sequence is simplicial, can be done just by testing the inclusion of a set in another one. The complexity is in the size of the biggest set. This size can be in $O(n)$, and the test can be done $O(n)$ times. Hence such an algorithm, using linear data structure, has a time complexity in $O(n^2)$. The time complexity can be improved, but then matrix should be used. And so the space complexity is not linear in this case.

We present now a sketch of the recognition algorithm, which works on the whole T. This algorithm can be seen like a parallel version of the previous idea. Since the procedure works on the whole tree T, all the inclusion tests in a given clique can be done at the same time. These inclusion tests are done by the procedures $test - simpliciality - first(C_i, S)$ and $test - simpliciality - next(C_i, S)$. These procedures correspond respectively to the lemmas 13, 14.

```
Algorithm 3: Recognition of Chordal Graph
Data: The result $T = (C, F)$ of the algorithm 2 as sorted lists
Result: Check whether $T$ is a clique-tree
begin
    for each node $C_i$ of $T$ do
        $S \leftarrow$ the set of sons of $C_i$ in $T$
        if not test-simpliciality-first($C_i, S$) then
            return $T$ is not a clique-tree, hence $G$ is not chordal
        if not test-simpliciality-next($C_i, S$) then
            return $T$ is not a clique-tree, hence $G$ is not chordal
    return $T$ is a clique-tree, hence $G$ is chordal
end
```

Let us study more precisely the complexity of both procedures $test - simpliciality$. The data structures are sorted lists respecting the elimination ordering. Each node of T is defined by two sorted lists: one for its vertex set, and the other for the corresponding increasing sequence. A sorted list of its mark set is associated to every vertex.

The procedure $test - simpliciality - first(C_i, S)$ checks at the same time, if the mark set of the first vertices of the increasing sequences corresponding to the sons of C_i are include in C_i. Since these sets are sorted lists, they are visited just once. Each list has a current element. The idea is to start at the beginning of each list. If all current element of the mark sets are less or equal to the current element of C_i, then the current element of C_i move to the following one. The current elements of the mark sets ore moved to the following if it is equal to the current element of C_i. If one of the current elements of the mark sets is smaller than the current element of C_i, then the test returns false. The end of the test is reached successfully if all the mark sets are visited before the end of C_i.

The procedure $test - simpliciality - next(C_i, S)$ checks whether the mark set of vertices, which are not the first one of an increasing sequence, are prefixes of the node containing it.

Theorem 15. *The recognition algorithm 3 checks whether G is chordal in linear time and space complexity $O(n + m)$.*

Proof: Since both procedures *test-simpliciality* correspond to the lemmas 13, 14, it is clear that the algorithm returns true if and only if the input T is a clique-tree.

The procedure 3 visits the data structure just once and always uses elementary operations. But in the previous section, we have shown that the size of a clique tree is $O(n + m)$. We can conclude that the space complexity of the algorithm 3 is $O(n + m)$. Therefore the time complexity is also $O(m + n)$. □

4 Conclusion

The clique graph of a chordal graph G introduced and studied in the first part, is shown to be the minimal structure containing all clique trees of G. The properties of the clique graph imply a simple algorithm for maximum spanning-trees which cannot be applied for general graphs. As in [3], MCS is compared to Prim's algorithm. Many versions of greedy algorithm for computing a maximal weighted independant set in a matroid are known [18, 11]. It could be worthwhile in this context to consider Kruskal's or Boruvka's alogrithm [4]. Hence, this matroidal approach can justify many versions of recognition algorithms of chordal graphs. It can be very helpful for various generalizations.

Hence this paper presents a new and unified regard on the MCS and BFS algorithms. The trace of these algorithms executions translates the behavior of the algorithm in the input graph. The study of this trace induces a simple way to compute the associated clique tree and the recognition of chordal graphs. This approach, studying the trace of algorithm like MCS and the underlying structure, can be very helpful for various generalizations of elimination orderings, see [13, 7], or generalizations of chordal graphs, see [10, 5].

Acknowledgement

We would like to thank J.X. Rampon for drawing our attention to references [3, 13] and R.H. Möhring for discussions on the BFS algorithm.

References

1. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. Assoc. Comput.*, 30:479–513, 1983.

2. C. Berge. *Hypergraphs.* North Hollands, 1989.
3. J.R.S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. preprint.
4. O. Boruvka. On a minimal problem. *Prac Moravske Predovedecke Spolecrosti*, 3, 1926.
5. A. Brandstädt, F.F. Dragan, V.D. Chepoi, and V.I Voloshin. Dually chordal graphs. In *Proceedings of the 19th Inter. Workshop on Graph-Theoretic Concept in Computer Science*, 1993. WG93.
6. P. Buneman. A characterization of rigid circuit graphs. *Discrete Math.*, 9:205–212, 1974.
7. E. Dahlhaus, P.L. Hammer, F. Maffray, and S. Olariu. On domination elimination orderings and domination graphs. Technical Report 27-94, Rutgers University Center of Operations Research, P.O. Box 5062, New Brunswick, New Jersey, USA, August 1994.
8. G.A. Dirac. On rigid circuit graphs. *Abh. Math. Sem. Uni. Hamburg 25*, 1961.
9. F. Gavril. The intersection graphs of a path in a tree are exactly the chordal graphs. *Journ. Comb. Theory*, 16:47–56, 1974.
10. Ryan B. Hayward. Weakly triangulated graphs. *Journal of Combinatorial theory*, 39:200–209, 1985. Serie B.
11. B. Korte, L. Lovász, and R. Schrader. *Greedoids.* Number 4 in Algorithms and Combinatorics. Springer Verlag, 1991.
12. J.G Lewis, B.W. Peyton, and A. Pothen. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput.*, 10(6):1146–1173, November 1989.
13. S. Olariu. Some aspects of the semi-perfect elimination. *Discrete Applied Mathematics*, 31:291–298, 1991.
14. B. Peyton. *Some applications of clique trees to the solutions of sparse linear systems.* PhD thesis, Clemson University, 1986.
15. R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 1957.
16. Donald J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysus and Applications*, 32:597–609, 1970.
17. Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal of Computing*, 5(2):266–283, June 1976.
18. P. Rosenstielh. L'arbre minimum d'un graphe. *Theory of Graphs*, 1967. P. Rosenstielh, editor, Gordon and Breach, New York.
19. Y. Shibata. On the tree representation of chordal graphs. *Journal of Graph Theory*, 12:421–428, 1988.
20. R.E. Tarjan and M. Yannakakis. Simple linear algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergaphs. *SIAM Journal of Computing*, 13:566–579, 1984.

This article was processed using the LaTeX macro package with LLNCS style