Organizing the atoms of the clique separator decomposition into an atom tree

Anne Berry, Romain Pogorelcnik, Geneviève Simonet

Organizing the atoms of the clique separator decomposition into an atom tree

Anne Berry* Romain Pogorelcnik† Geneviève Simonet‡

April 22, 2014

Abstract

We define an atom tree of a graph as a generalization of a clique tree: its nodes are the atoms obtained by clique minimal separator decomposition, and its edges correspond to the clique minimal separators of the graph.

Given a graph G, we compute an atom tree by using a clique tree of a minimal triangulation H of G. Computing an atom tree with such a clique tree as input can be done in $O(\min(nm, m + nf))$, where f is the number of fill edges added by the triangulation. When both a minimal triangulation and the clique minimal separators of G are provided, we compute an atom tree of G in $O(m + f)$ time, which is in $O(n^2)$ time.

We give an $O(nm)$ time algorithm, based on MCS, which combines in a single pass the 3 steps involved in building an atom tree: computing a minimal triangulation, constructing a clique tree, and constructing the corresponding atom tree.

Finally, we present a process which uses a traversal of a clique tree of a minimal triangulation to determine the clique minimal separators and build the corresponding atom tree in $O(n(n + t))$ time, where t is the number of 2-pairs of H (t is at most $\overline{m} - f$, where $\overline{m}$ is the number of edges of the complement graph); to complete this, we also give an algorithm which computes a minimal triangulation in $O(n(n + \overline{m}))$ time, thus providing an approach to compute the decomposition in $O(n(n+\overline{m}))$ time.

keywords: clique separator decomposition, minimal triangulation, clique tree, atom tree, MCS.

1 Background and motivation

Clique separator decomposition was introduced by Tarjan [44] as a useful hole- and C_4- preserving decomposition, which enables a possible Divide-and-Conquer

*LIMOS UMR CNRS 6158, Ensemble Scientifique des Cézeaux, F-63 173 Aubière, France; research partially supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010; berry@isima.fr

†Laboratoire Génétique Reproduction et Développement, Université d'Auvergne, Institut National de la Santé et de la Recherche Médicale, UMR 1103; Centre National de la Recherche Scientifique, UMR 6293; romain.pogorelcnik@udamail.fr

‡LIRMM UMR 5506, 161 Rue Ada, F-34392 Montpellier, France; simonet@lirmm.fr

approach to several hard problems such as Minimum Fill-in, Maximum Clique, Coloring, and Maximum Weighted Independent Set, as these problems can be solved separately on each of the subgraphs resulting from the decomposition. (A *clique separator* is a clique whose removal increases the number of connected components of the graph.)

This decomposition consists in repeatedly finding a clique separator S of a graph G and a partition of $G - S$ into two subgraphs A and B such that no vertex of A is adjacent to a vertex of B, and dividing the graph into the two subgraphs $A \cup S$ and $B \cup S$; this step is repeated until none of the subgraphs obtained contains a clique separator. The resulting subgraphs are called *atoms*.

Tarjan [44] improved the complexity for finding a clique separator proposed by Whitesides [46] from $O(n^3)$ per clique separator to $O(nm)$ for all the clique separators used in a decomposition. The breakthrough which enables Tarjan to do this is the proof of the strong relationship between the clique separators of a graph and minimal triangulation. (*Minimal triangulation* is the process of embedding a graph into a chordal graph by the addition of an inclusion-minimal set of *fill* edges.) He showed that no fill edge of a minimal triangulation can have its extremities in two different components defined by a clique separator.

Tarjan's algorithm was a two-step process: he first computed a minimal triangulation in $O(nm)$ time, then performed n graph searches to find the decomposition in $O(nm)$ time.

Tarjan [44] left open the question of refining the decomposition to obtain a unique set of atoms. This was solved by Leimer [36], who showed that when clique *minimal* separators are used, the set of atoms becomes unique, and is exactly the set of maximal connected subgraphs with no clique separator. (Minimal separators are defined in Section 2.) Leimer showed that all the clique minimal separators of a graph G are minimal separators of any minimal triangulation H of G. This makes it easy to find the clique minimal separators, since there are few minimal separators in a chordal graph. Leimer streamlined the algorithm from [44] accordingly, but he used the same two-step approach as Tarjan, and did not improve the time complexity for the decomposition.

Recently, clique separator decomposition has generated interest in several fields. It has been used on special graph classes to solve the Maximum Weight Stable Set Problem [20, 21], and for recognition purposes [19, 8]. It has also been applied to treewidth [18] and to Bayesian networks to help compute a better triangulation [39]. In the field of bioinformatics, the clique minimal separators help yield evolutionary information [31], and the atoms of the decomposition yield interesting gene clusters [32]. The reader is also referred to [13] for a survey on this decomposition. Another related result is by Kratsch and Spinrad [34], where they show that determining whether a graph on $4n + 2$ vertices has a clique separator is at least as hard as determining whether an n-vertex graph has a simplicial vertex, even if a minimal elimination ordering is given as part of the input.

In the past few years, minimal triangulation has also given rise to renewed interest. Several papers provide new $O(nm)$ time algorithms [3, 7, 10] (see also the special issue on the subject [4], and the survey therein [27]). Efforts have been invested into improving this $O(nm)$ time bound for dense graphs: Kratsch and Spinrad [33] offer an $O(n^{2.69})$ time bound, improved by Heggernes, Telle and Villanger [28] to $O(n^{\alpha}\log n)$, where n^{α} is the time required to do matrix multiplication. Another approach is to compute a non-minimal triangulation

and then remove any extraneous edges: Blair, Heggernes and Telle [16] present an $O(f(m+f))$ time algorithm, where f is the number of fill edges, and which is efficient when f is small; Mezzini and Moscarini [38] give an $O(\overline{m}(\delta^2+\overline{m}))$ time algorithm, where $\overline{m}$ is the number of edges of $\overline{G}$ and δ is the maximum degree of $\overline{G}$.

For special graph classes, several results have appeared, which improve on the time bound for the general case for computing a minimal triangulation: chordal bipartite graphs and hole-and-diamond-free graphs have been shown to have an $O(n^2)$ time algorithm [8]; co-comparability graphs and AT-free claw-free graphs have a linear time algorithm [37]; co-bipartite graphs, a subclass of AT-free claw-free graphs, have an even better time, since only a subset of edges needs to be traversed [14]; claw-free graphs with paths of bounded length can be triangulated in linear time [15].

However, no improvement has been proposed to decrease the time to compute clique minimal separator decomposition to match these advances in minimal triangulation. In most of the above-mentioned algorithms, the clique minimal separators of the graph are not found directly by the triangulation process, so they have to be computed separately; but even when the minimal triangulation and the clique minimal separators are given as input, computing the decomposition still takes at least time proportional to nm, as up to n graph searches still need to be performed in all known algorithms.

Our ultimate goal is to open an area for improvement on the time required to obtain the decomposition once the minimal triangulation H of the input graph G is computed. More specifically, we want to bypass the up to n graph searches which will cost $O(nm)$ time.

With this idea in mind, we introduce a tree structure, the *atom tree*, which organizes the atoms of the clique minimal separator decomposition into a tree, in much the same way as the clique tree organizes the maximal cliques of a chordal graph. In fact, the atom tree is a tree whose nodes are the atoms and whose edges correspond to the clique minimal separators of the graph, just as the clique tree is a tree whose nodes are the maximal cliques and whose edges correspond to the minimal separators of the chordal graph. (The atom trees of a chordal graph are exactly its clique trees.) One of the strong points of this atom tree is that it yields a visualization of the way the atoms of the graph are structured, but in our view it is also a promising data structure which may in the future yield improvements in the computation of the clique separator decomposition once a clique tree of a minimal triangulation is provided.

In this paper, we introduce the concept of atom tree, and explore some of its properties. We provide a simple algorithmic process which uses a clique tree of a minimal triangulation H of the input graph G to compute an atom tree of G. When the clique minimal separators of G are given in the input along with the minimal triangulation, we compute an atom tree of G in $O(m+f)$ time, where $m+f$ is the number of edges of H. When the clique minimal separators are not provided, our basic process costs $O(\min(nm, m+nf))$ time.

We can thus compute an atom tree in three steps. The first step computes a minimal triangulation H of the input graph G; the second step constructs a clique tree T of H; and the third step modifies T into the corresponding atom tree. We devise a process which combines these three steps into a single $O(nm)$-time algorithm, by merging several variants of MCS, as follows. Computing a minimal triangulation H of G can be done with Algorithm MCS-M from [3],

and yields an MCS ordering of the vertices of H; the algorithm from [17] uses an MCS ordering of a chordal graph to compute a clique tree and to define the minimal separators; we are thus able to merge these two algorithms, and to add a loop which tests the minimal separators as they are found, in order to determine which are clique minimal separators of G, and process them accordingly. The algorithm is easy to implement.

We then investigate how we can make use of the tree structure of the clique tree T of H to derive some improvements on the time complexity. We introduce a process which traverses T and converts it into the corresponding atom tree of G in $O(n(n+t))$ time, where t is the number of 2-pairs of H. (A 2-pair is a pair of non-adjacent vertices such that every chordless path from one vertex to the other is of length 2, so $t \leq \overline{m}$); this approach can probably be improved upon to yield a better time bound. We complement this result with an algorithm to compute a minimal triangulation in $O(n(n+\overline{m}))$ time, improving the result from [38], which makes the complexity of computing an atom tree from the input graph in $O(n(n+\overline{m}))$ time.

The paper is organized as follows: in Section 2, we provide the necessary graph notions; in Section 3, we define the notion of atom tree and prove properties which will enable us to compute it, then provide the corresponding generic algorithm. In Section 4, we give our algorithm based on MCS to compute the atom tree directly from the input graph. In Section 5, we present our process to compute an atom tree from a minimal triangulation in $O(n(n+t))$ time, and we explain how to compute a minimal triangulation of a graph in $O(n(n+\overline{m}))$ time, before concluding.

2 Preliminaries

Basics.
All graphs in this work are connected, undirected and finite. A graph is denoted by $G = (V, E)$, with $|V| = n$ and $|E| = m$. $\overline{G} = (V, \overline{E})$ will denote the complement of graph $G = (V, E)$, with $\overline{m} = |\overline{E}|$. The *neighborhood* of a vertex x in a graph G is $N_G(x)$, or $N(x)$ if the context is clear; the *closed neighborhood* of x is $N[x] = N(x) \cup \{x\}$. The neighborhood of a subset X of V is $N(X) = (\cup_{x \in X} N(x)) - X$. A *2-pair* of a connected graph G is a pair $\{x, y\}$ of vertices of G such that every chordless path between x and y is of length 2. A *clique* is a set of pairwise adjacent vertices (a single vertex is also a clique); we say that we *saturate* a set X of vertices when we add all the edges necessary to turn X into a clique. A vertex is *simplicial* if its neighborhood is a clique. A *clique module* is a set X of vertices such that $\forall x, y \in X, N[x] = N[y]$. Graph $G(X)$ denotes the subgraph induced by the set X of vertices, but we will sometimes just denote this by X. The reader is referred to [26] and [22] for classical graph definitions and results.

Separators.
A set S of vertices of a connected graph G is a *separator* if $G(V - S)$ is not connected. A separator S is an *xy-separator* if x and y lie in two different connected components of $G(V - S)$. A separator S is a *minimal xy-separator* if no proper subset of S is also an xy-separator. A separator S is said to be *minimal* if there are two vertices x and y such that S is a minimal xy-separator.

Equivalently, S is a minimal separator if and only if $G(V-S)$ has at least two connected components C_1 and C_2 such that $N_G(C_1) = N_G(C_2) = S$. A *clique minimal separator* is a minimal separator which is a clique. There are less than n clique minimal separators in a graph. A pair $\{x, y\}$ of vertices is a 2-pair of G if and only if $N(x) \cap N(y)$ is a minimal xy-separator of G [1]. A *moplex* is a clique module M whose neighborhood $N(M)$ is a minimal separator.

Chordal graphs and minimal triangulations.
A graph is *chordal* (or triangulated) if it contains no chordless induced cycle of length 4 or more. A chordal graph has at most n maximal cliques, and the sum of their sizes is at most $n + m$. A graph is chordal if and only if all its minimal separators are cliques [24]. A chordal graph has at most n minimal separators [42].

A chordal graph is often represented by a *clique tree*:

Definition 2.1 *Let $H = (V, E)$ be a connected chordal graph. A* clique tree *of H is a tree $T = (V_T, E_T)$ such that V_T is the set of maximal cliques of H and for any vertex x of H, the subgraph T_x of T induced by the set of nodes of T containing x is a subtree of T.*

Property 2.2 *[17] Let H be a connected chordal graph, let T be a clique tree of H, and let S be a set of vertices of H; then S is a minimal separator of H if and only if there is an edge K_1K_2 of T such that $S = K_1 \cap K_2$.*

Every chordal graph has at least one clique tree, which can be computed in linear time with the nodes labeled by the maximal cliques and the edges labeled by the minimal separators [17].

Given a non-chordal graph $G = (V, E)$, the supergraph $H = (V, E + F)$ is a *triangulation* of G if H is chordal. F is the set of *fill edges*, $|F|$ is denoted by f. The triangulation is *minimal* if for any proper subset of edges $F' \subset F$, the graph $(V, E + F')$ fails to be chordal.

Peos and meos.
An ordering of a graph $G = (V, E)$ is a one-to-one mapping from $\{1, \ldots, n\}$ to V. Given an ordering α of G, the graph $G^+_\alpha = (V, E + F_\alpha)$ is defined as follows: initialize the current graph G' with G and the set F_α with the empty set, then for each i from 1 to n, add the set F_i of edges necessary to saturate the neighborhood of $\alpha(i)$ in G', add the edges of F_i to F_α and remove $\alpha(i)$ from G'. $G^+_\alpha = (V, E + F_\alpha)$ is a triangulation of G. α is called a *perfect elimination ordering (peo)* of G if $F_\alpha = \emptyset$; in this case, α is a simplicial elimination ordering (at each step, the eliminated vertex is simplicial in G'), and the graph is chordal: a graph is chordal if and only if it has a peo [25]. α is called a *minimal elimination ordering (meo)* of G if G^+_α is a minimal triangulation of G. Given an ordering α of a graph $G = (V, E)$ and a vertex x of V, $Madj_G(x)$ denotes the set of neighbors of x which have a number greater than that of x with respect to α [17]. In a chordal graph H with a peo α, vertex x is called a *minimal separator generator* when $Madj_H(x)$ is a minimal separator of H [12].

A *moplex ordering* of a chordal graph G is a partition $\{X_1 \ldots X_k\}$ of V into cliques such that for any $i \in \{1 \ldots k-1\}$, X_i is a moplex of graph G_i obtained from G by removing the vertices of $X_1, \ldots, X_{i-1}$. Algorithms LexBFS from [43] and MCS from [45] compute a moplex ordering of a chordal graph,

and algorithms LEX M from [43] and MCS-M from [3] compute a minimal triangulation and a moplex ordering of this minimal triangulation [2].

Clique minimal separator decomposition and atoms.
Decomposing a connected graph $G = (V, E)$ by its clique minimal separators can be done by repeatedly finding a clique minimal separator S of G, with $\{C_1 \dots C_k\}$ the connected components of $G(V - S)$, and replacing G with the set of subgraphs $G(C_i \cup N(C_i))$, with $i \in [1, k]$ [13].

The subgraphs obtained in the end, which we call *atoms*, are the maximal connected subgraphs of G having no clique separator [36]. In this paper, we will refer to atoms indifferently as subgraphs or as vertex sets. A graph has at most n atoms. The atoms of a chordal graph are its maximal cliques.

Finding the clique minimal separators of a graph can be done by first computing a minimal triangulation, as described by the following property:

Property 2.3 *[36][41][13] Let $G = (V, E)$ be a connected graph, and let $H = (V, E + F)$ be a minimal triangulation of G. Any minimal separator S of H is a minimal separator of G, the connected components of $G(V - S)$ are the same as those of $H(V - S)$, and for each such component C we have that $N_H(C) = N_G(C)$. The clique minimal separators of G are the minimal separators of H that are cliques in G.*

The reader is referred to [13] for full details on this decomposition.

Elementary decompositions.
We call *elementary decomposition by clique minimal separator*, or *elementary decomposition* for short, of a connected graph $G = (V, E)$ a partition (Y_1, S, Y_2) of V such that S is a clique of G, no vertex of Y_1 is adjacent to a vertex of Y_2 in G and each one of Y_1 and Y_2 contains a component of $G(V - S)$ whose neighborhood is S. Note that in that case $G(Y_1 \cup S)$ and $G(Y_2 \cup S)$ are connected.

To prove that for each edge K_1K_2 of a clique tree T of a chordal graph H, $K_1 \cap K_2$ is a minimal separator of H, [17] shows the following result.

Property 2.4 *[17] Let H be a connected chordal graph, let T be a clique tree of H, let K_1K_2 be an edge of T, let $S = K_1 \cap K_2$, and for each $i \in \{1, 2\}$ let T_i be the connected component of $T - \{K_1K_2\}$ containing K_i and let V_i be the union of the nodes of T_i; then $(V_1 - S, S, V_2 - S)$ is an elementary decomposition of H.*

This elementary decomposition can be used as an elementary step in clique minimal separator decomposition. Thus a decomposition step can consist of finding an elementary decomposition (Y_1, S, Y_2) of the current graph $G' = (V', E')$ and replacing V' by $Y_1 \cup S$ and $Y_2 \cup S$, which corresponds to the following property:

Property 2.5 *[36] Let G be a connected graph, and let (Y_1, S, Y_2) be an elementary decomposition of G; then the set of atoms of G is the disjoint union of the sets of atoms of $G(Y_1 \cup S)$ and of $G(Y_2 \cup S)$.*

3 Computing an atom tree of a graph

3.1 Defining atom trees

To represent the atoms obtained by clique minimal separator decomposition, we extend the notion of clique tree of a chordal graph to that of *atom tree* of an arbitrary graph:

Definition 3.1 *Let $G = (V, E)$ be a connected graph. An* atom tree *of G is a tree $T = (V_T, E_T)$ such that V_T is the set of atoms of G and for any vertex x of G, the subgraph T_x of T induced by the set of nodes of T containing x is a subtree of T.*

As is the case for a clique tree, the intersection of two adjacent nodes of an atom tree is a clique minimal separator.

Property 3.2 *Let G be a connected graph, let T be an atom tree of G, and let S be a set of vertices of G; then S is a clique minimal separator of G if and only if there is an edge A_1A_2 of T such that $S = A_1 \cap A_2$.*

To prove Property 3.2, we will use the graph G^* introduced by [36]:

Property 3.3 *[36] Let $G = (V, E)$ be a connected graph, and let G^* be the graph obtained from G by saturating its atoms; then G^* is chordal, its maximal cliques are the atoms of G and for any subsets S, Y_1 and Y_2 of V, S is a clique minimal separator of G if and only if it is a minimal separator of G^*, and (Y_1, S, Y_2) is an elementary decomposition of G if and only if it is an elementary decomposition of G^*.*

Proof: (of Property 3.2) By Property 3.3, T is a clique tree of G^*. S is a clique minimal separator of G if and only if it is a minimal separator of G^* by Property 3.3, *i.e.* if and only if there is an edge A_1A_2 of T such that $S = A_1 \cap A_2$ by Property 2.2. □

An atom tree not only yields the atoms of the decomposition by clique minimal separators, but also provides a compact representation of the organization of the atoms, as illustrated by Figure 1.

Example 3.4 *Figure 1 shows a graph G and one of its atom trees. Each edge A_1A_2 of the atom tree is labeled with the associated clique minimal separator $A_1 \cap A_2$ of G (G has 2 atom trees, as node $\{h, j, k\}$ can be adjacent to each of the two other nodes containing h).*

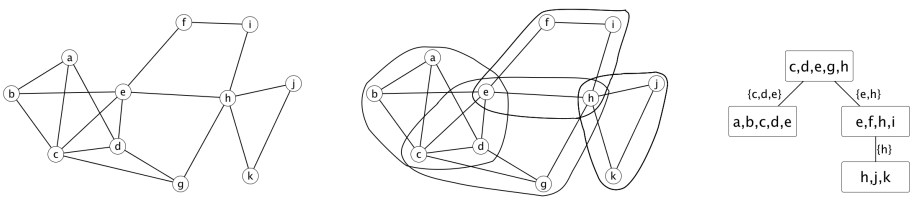

Figure 1: Graph G, the atoms of G, and an atom tree of G.

To compute an atom tree from a clique tree of a minimal triangulation, we will use a merging technique which contracts all the nodes of the clique tree that are not separated by an edge representing a clique minimal separator of the input graph:

Theorem 3.5 *Let $G = (V, E)$ be a connected graph, let $H = (V, E + F)$ be a minimal triangulation of G, let $T = (V_T, E_T)$ be a clique tree of H, and let T' be the forest obtained from T by removing all edges KK' such that $K \cap K'$ is a clique in G, let T'' be the tree obtained from T by merging the nodes of each tree of T' into one node; then T'' is an atom tree of G, and for each edge KK' of T such that $K \cap K'$ is a clique in G, $K \cap K' = A \cap A'$, where A and A' are the atoms of G containing K and K' respectively.*

To prove Theorem 3.5, we will extend Property 2.4 to atom trees:

Property 3.6 *Let G be a connected graph, let T be an atom tree of G, let A_1A_2 be an edge of T, let $S = A_1 \cap A_2$, and for each $i \in \{1, 2\}$ let T_i be the connected component of $T - \{A_1A_2\}$ containing A_i, let V_i be the union of the nodes of T_i and let $G_i = G(V_i)$; then $(V_1 - S, S, V_2 - S)$ is an elementary decomposition of G and for each $i \in \{1, 2\}$, T_i is an atom tree of G_i.*

Proof: (of Property 3.6)
By Property 3.3, T is a clique tree of G^*. By Property 2.4, $(V_1 - S, S, V_2 - S)$ is an elementary decomposition of G^*, so by Property 3.3, it is an elementary decomposition of G. It follows by Property 2.5 that the set of atoms of G is the disjoint union of the sets of atoms of G_1 and of G_2. Let $i \in \{1, 2\}$, and let us show that T_i is an atom tree of G_i. As for each vertex x of G_i, $(T_i)_x$ is a subtree of T_i (since it is the restriction of T_x to T_i), it is sufficient to show that the nodes of T_i are the atoms of G_i. The set of atoms of G is the disjoint union of the sets of nodes of T_1 and of T_2, and is also the disjoint union of the sets of atoms of G_1 and of G_2. As moreover no atom of G_1 can be a subset of V_2 and no atom of G_2 can be a subset of V_1 (otherwise it would be a subset of $V_1 \cap V_2 = S$, and therefore a strict subset of A_1 and A_2) the atoms of G_i are exactly the nodes of T_i. □

Conversely, we can build an atom tree of G from atom trees of subgraphs of G.

Property 3.7 *Let G be a connected graph, let (Y_1, S, Y_2) be an elementary decomposition of G and for each $i \in \{1, 2\}$ let T_i be an atom tree of $G(Y_i \cup S)$ and let A_i be a node of T_i containing S; then the tree T composed of T_1, T_2 and edge A_1A_2 is an atom tree of G.*

Proof: (of Property 3.7)
The set of nodes of T is the disjoint union of the sets of nodes of T_1 and T_2, which are the sets of atoms of $G(Y_1 \cup S)$ and of $G(Y_2 \cup S)$. It follows by Property 2.5 that the nodes of T are the atoms of G. Moreover, for each vertex x of G, T_x is a subtree of T: if $x \in S$ then T_x is the tree composed of $(T_1)_x$, $(T_2)_x$ and the edge A_1A_2, otherwise it is either equal to $(T_1)_x$ or to $(T_2)_x$. Hence T is an atom tree of G. □

Proof: (of Theorem 3.5)
We prove this by induction on the number of edges KK' of T such that $K \cap K'$ is a clique in G. If there is no such edge then by Properties 2.2 and 2.3, G has no clique minimal separator and therefore V is the unique atom, and T'' is reduced to node V, so the property holds. We assume that it holds if T has at most k edges KK' such that $K \cap K'$ is a clique in G. Let us show that it holds if T has $k+1$ such edges.

Let K_1K_2 be an edge of T such that $K_1 \cap K_2$ is a clique in G, let $S = K_1 \cap K_2$, and for each $i \in \{1,2\}$ let T_i be the connected component of $T - \{K_1K_2\}$ containing K_i, let T_i'' be the tree obtained from T_i by the merging process, let V_i be the union of the nodes of T_i, let $G_i = G(V_i)$ and let $H_i = H(V_i)$.

By Property 2.4, $(V_1 - S, S, V_2 - S)$ is an elementary decomposition of H. By Property 2.3, this is also the case for G.

Let us show that H_1 is a minimal triangulation of G_1 and that T_1 is a clique tree of H_1. H_1 is clearly a triangulation of G_1. We claim that this triangulation is minimal: if it is not, let H' be a triangulation of G_1 that is a strict subgraph of H_1; the union of H' and H_2 is a triangulation of G that is a strict subgraph of H, which contradicts the minimality of H. As T is a clique tree of H, it is also an atom tree of H, so, by Property 3.6, T_1 is an atom tree of H_1, and therefore a clique tree of H_1 since H_1 is chordal. By induction hypothesis, T_1'' is an atom tree of G_1 and for each edge KK' of T_1 such that $K \cap K'$ is a clique in G, $K \cap K' = A \cap A'$, where A and A' are the atoms of G_1 containing K and K' respectively.

By symmetry, this is also the case for T_2'' and G_2. As $S = K_1 \cap K_2 \subseteq A_1 \cap A_2$, by Property 3.7, T'' is an atom tree of G.

For A_1 and A_2 the atoms of G containing K_1 and K_2 respectively, we have $S \subseteq A_1 \cap A_2 \subseteq V_1 \cap V_2 = S$, and therefore $S = A_1 \cap A_2$, which completes our proof. □

Example 3.8 *Figure 2 shows a minimal triangulation H of graph G from Figure 1 (fill edges are dashed), a clique tree T of H (minimal separators which are cliques in G are prefixed with a star) and the atom tree of G obtained from T by the merging process.*

3.2 Computing an atom tree from a clique tree of a minimal triangulation

From Theorem 3.5, we derive an algorithm to compute an atom tree from a clique tree of a minimal triangulation. Our algorithm defines the connected components of the forest T' by contracting each pair of adjacent nodes in T separated by an edge which does not represent a clique minimal separator (the *contraction* of two nodes K and K' into a single node consists in replacing K and K' with node $K \cup K'$; the set of its neighbors in the new tree is the union of the sets of neighbors of K and of K').

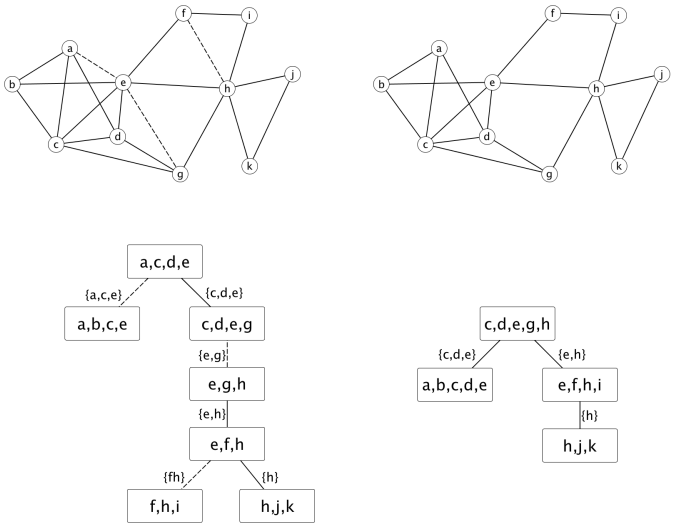

Figure 2: Graph G and a minimal triangulation H of G *(fill edges are dashed)*, a clique tree T of H *(the minimal separators which are not cliques in G are represented by dashed edges in the clique tree)*, and the atom tree of G obtained from T by the merging process.

Algorithm Atom-Tree

input : A connected graph G, a clique tree T of a minimal triangulation of G.
output: T is merged into the corresponding atom tree of G.
foreach *edge* KK' *of* T **do**
  if $K \cap K'$ *is not a clique in* G **then**
    Contract K with K' ;

Theorem 3.9 *Let $G = (V, E)$ be a graph, and let $H = (V, E + F)$ be a minimal triangulation of G, with $|F| = f$. Algorithm Atom-Tree computes an atom tree of G from a clique tree of H in time:*

- $O(m + f)$ *if the clique minimal separators of G are provided.*
- $O(\min(nm, m + nf))$ *otherwise.*

Proof: As T is a tree, it can be searched from one of its nodes in such a way that for each processed edge KK', K' is still unreached, and therefore is a node of the initial tree T with the same degree as in T. Computing $K \cap K'$ and contracting K with K' can be done in $O(|K'| + |N_T(K')|)$ time, hence in $O(m + f)$ time globally since the sum of the sizes of the nodes of T is at most $n + m + f$ and T has less than n edges.

If on the one hand the set of clique minimal separators of the input graph is provided, determining whether $K \cap K'$ is one of them costs $O(m + f)$ time globally: we build the tripartite graph $I = (V, E_T, \mathscr{S}^*, E_I)$, where E_T is the set of edges of the clique tree, $\mathscr{S}^*$ is the set of clique minimal separators of G,

and E_I is the set of edges of the tripartite graph, defined as follows: $vy \in E_I$ if $v \in V$ and y is either a minimal separator of $\mathscr{S}^*$ containing v or an edge of the clique tree representing a minimal separator containing v. I has less than $3n$ vertices and less than $2(n+m+f)$ edges, since there are at most n minimal separators in H and since the sum of the sizes of the minimal separators of H, which is less than the sum of the sizes of its maximal cliques, is at most $n+m+f$. In I, we search for pairs of vertices $\{x, y\}$, $x \in \mathscr{S}^*, y \in E_T$, with identical neighborhoods. Finding sets of vertices with identical neighborhoods can be done in linear time using partition refinement on the neighborhoods [30].

If on the other hand the clique minimal separators of G are not provided, determining whether $K \cap K'$ is a clique in G costs $O(m)$ time, and therefore $O(nm)$ time globally. Another way to determine whether $K \cap K'$ is a clique in G is to check for each fill edge whether it belongs to $K \cap K'$, which can be done in $O(f)$ time per edge of T, hence in $O(nf)$ time globally, which makes the total complexity in $O(m+nf)$ time. □

The time complexity in $O(m+nf)$ may be better than $O(nm)$ when f is of small size. This is interesting in the context of triangulations computed with the Minimum Degree Heuristic, which approximates minimum triangulation and can yield a small fill in practice [16].

When $O(nm)$ is better than $O(m+nf)$, our algorithm runs faster than in the general case when the sum of the number of edges in G over all the minimal separators of H is small, which happens when the separators are of small size, when the overlap between separators contains few edges globally, when there are few separators, or when there are few maximal cliques. This of course is also the case when the size of the maximum clique of H is bounded.

4 Computing an atom tree directly from the input graph

In this section, we explain how we can compute an atom tree directly from the input graph in a single pass by combining two variations of Algorithm MCS.

MCS [45] was devised as a simplification of Algorithm LexBFS [43] to compute a peo α of a chordal graph H. Originally, MCS (for Maximum Cardinality Search), numbers the vertices from n to 1 and is described as using weights (or labels) on unnumbered vertices. The label of a vertex x corresponds to the number of already numbered neighbors of x, which is the number $|Madj_H(x)|$ of neighbors of x which have a higher number than that of x if x is the vertex being numbered.

The first modification of MCS, called *Expanded MCS*, was proposed by Blair and Peyton [17] to build a clique tree of a chordal graph H in linear time: MCS is run (from n to 1), and each time the vertex x which is chosen to be numbered has a label which is not strictly larger than the label of the previously chosen vertex, a maximal clique has just been completed, x is a minimal separator generator, and an edge can be added to the clique tree, labeled with $Madj_H(x)$.

The second modification of MCS is Algorithm MCS-M [3], which computes a minimal triangulation H of a non-chordal graph G, as well as an meo α of G, in $O(nm)$ time. MCS-M extends MCS in the same fashion that LEX M extends

LexBFS in [43]: a feature computing the fill edges incident to the current vertex is added.

The ordering α which is computed by MCS-M is both an meo of G and a peo of the computed minimal triangulation H [2]; using the idea behind Expanded MCS, a clique tree of the minimal triangulation can therefore be computed at the same time as the ordering. We thus define *Algorithm Expanded MCSM*.

Finally, using the results from Section 3.2, we are able to modify Expanded MCSM so that it uses only minimal separators of the minimal triangulation which are also clique minimal separators of the input graph. This yields Algorithm MCSM-Atom-Tree, which is a combination of Expanded MCS, MCS-M and Algorithm Atom-Tree, and which computes an atom tree of a non-chordal graph in a single $O(nm)$-time pass.

To make the paper self-contained, we will first recall algorithms MCS, Expanded MCS, and MCS-M, before presenting Expanded MCSM and finally MCSM-Atom-Tree.

In the following algorithms, V' is the set of not yet numbered vertices; $|N_H(x) - V'|$ is the label of vertex x, which is equal to $|Madj_H(x)|$ if x is the vertex being numbered.

A clique tree of a chordal graph H is represented by a labeled tree $T = (V_T, E_T, Clique, MinSep)$, where V_T is in the form $\{1 \dots q\}$, $Clique$ maps each node of V_T to a maximal clique of H and $MinSep$ maps each edge of E_T to a minimal separator of H. For each edge ps in E_T, $MinSep(ps) = Clique(p) \cap Clique(s)$ is the corresponding minimal separator.

Similarly, an atom tree of a graph G is represented by a labeled tree $T = (V_T, E_T, Atom, CliqueMinSep)$, where V_T is in the form $\{1 \dots q\}$, $Atom$ maps each node of V_T to an atom of G and $CliqueMinSep$ maps each edge of E_T to a clique minimal separator of G, with $CliqueMinSep(ps) = Atom(p) \cap Atom(s)$.

Algorithm MCS[45]

input : A chordal graph $H = (V, E)$.
output: A peo α of H.
$V' \leftarrow V$;
for *i from n downto* 1 **do**
  Choose a vertex x in V' such that $|N_H(x) - V'|$ is maximum;
  $\alpha(i) \leftarrow x$; $V' \leftarrow V' - \{x\}$;

Algorithm Expanded-MCS[17]

input : A connected chordal graph $H = (V, E)$.
output: A peo α, the corresponding clique tree
$T = (V_T, E_T, Clique, MinSep)$ of H.

$V' \leftarrow V$; *prev-card* $\leftarrow 0$; $s \leftarrow 0$; $V_T \leftarrow \emptyset$; $E_T \leftarrow \emptyset$;
for *i from n downto* 1 **do**
  Choose a vertex x in V' such that $|N_H(x) - V'|$ is maximum;
  $\alpha(i) \leftarrow x$; *new-card* $\leftarrow |N_H(x) - V'|$ // $= |Madj_H(x)|$;
  if *new-card* $\leq$ *prev-card* *//begin new clique* **then**
    $s \leftarrow s + 1$; $V_T \leftarrow V_T + \{s\}$; $Clique(s) \leftarrow Madj_H(x)$;
    if *new-card* $\neq 0$ *//get edge to parent* **then**
      $k \leftarrow \min\{j \mid \alpha(j) \in Madj_H(x)\}$;
      $p \leftarrow cli(\alpha(k))$;
      $E_T \leftarrow E_T + \{ps\}$;
      $MinSep(ps) \leftarrow Madj_H(x)$ *//x is a separator generator*;
  $cli(x) \leftarrow s$: $Clique(s) \leftarrow Clique(s) + \{x\}$; $V' \leftarrow V' - \{x\}$;
  prev-card $\leftarrow$ *new-card*;

Algorithm MCS-M[3]

input : A graph $G = (V, E)$.
output: A minimal triangulation $H = (V, E + F)$ of G, the
corresponding meo α of G, which is a peo of H.

$E_H \leftarrow E$; $H \leftarrow (V, E_H)$; $F \leftarrow \emptyset$; $V' \leftarrow V$;
for *i from n downto* 1 **do**
  Choose a vertex x in V' such that $|N_H(x) - V'|$ is maximum;
  $\alpha(i) \leftarrow x$;
  foreach $y \in V' - N_H[x]$ **do**
    if *there is an xy-path* μ *in* $G(V')$ *such that for each internal node* z *of* μ $|N_H(z) - V'| < |N_H(y) - V'|$ **then**
      $F \leftarrow F + \{xy\}$; $E_H \leftarrow E_H + \{xy\}$;
  $H \leftarrow (V, E_H)$; $V' \leftarrow V' - \{x\}$;

Combining Algorithms Expanded MCS and MCS-M, we obtain the following algorithm:

Algorithm Expanded-MCSM

input : A connected graph $G = (V, E)$.
output: A minimal triangulation $H = (V, E + F)$ of G, an meo α of G (α is a peo of H), and the corresponding clique tree $T = (V_T, E_T, Clique, MinSep)$ of H.

$E_H \leftarrow E$; $H \leftarrow (V, E_H)$; $F \leftarrow \emptyset$; $V' \leftarrow V$; $s \leftarrow 0$; $V_T \leftarrow \emptyset$; $E_T \leftarrow \emptyset$;
for i *from* n *downto* 1 **do**
  Choose a vertex x in V' such that $|N_H(x) - V'|$ is maximum;
  $\alpha(i) \leftarrow x$; *new-card* $\leftarrow |N_H(x) - V'|$ // $= |Madj_H(x)|$;
  if *new-card* $\leq$ *prev-card* **then**
    $s \leftarrow s + 1$; $V_T \leftarrow V_T + \{s\}$; $Clique(s) \leftarrow Madj_H(x)$;
    if *new-card* $\neq 0$ **then**
      $k \leftarrow \min\{j \mid \alpha(j) \in Madj_H(x)\}$; $p \leftarrow cli(\alpha(k))$;
      $E_T \leftarrow E_T + \{ps\}$; $MinSep(ps) \leftarrow Madj_H(x)$;
  $cli(x) \leftarrow s$: $Clique(s) \leftarrow Clique(s) + \{x\}$;
  foreach $y \in V' - N_H[x]$ **do**
    if *there is an* xy*-path* μ *in* $G(V')$ *such that for each internal node* z *of* μ $|N_H(z) - V'| < |N_H(y) - V'|$ **then**
      $F \leftarrow F + \{xy\}$; $E_H \leftarrow E_H + \{xy\}$;
  $H \leftarrow (V, E_H)$; $V' \leftarrow V' - \{x\}$; *prev-card* $\leftarrow$ *new-card*;

We will now modify Algorithm Expanded-MCSM to compute the corresponding atom tree directly.

For each edge ps of the clique tree of the minimal triangulation H computed by an execution of Algorithm Expanded-MCSM, the minimal separator $Clique(p) \cap Clique(s)$ of H is $Madj_H(x)$, where x and H are the values of these variables when adding edge ps to E_T. Thus, in order to compute an atom tree of a graph G, we will modify Algorithm Expanded-MCSM as follows: when *new-card* $\leq$ *prev-card*, we determine whether $Madj_H(x)$ is a clique in G; if it is, we increment s, and add edge ps to E_T if *prev-card* $\neq 0$; if it is not, we merge $Clique(p)$ and $Clique(s)$ by setting s to p. We obtain the following algorithm.

Algorithm MCSM-Atom-Tree

input : A connected graph $G = (V, E)$.
output: An atom tree $T = (V_T, E_T, Atom, CliqueMinSep)$ of G, a set F of fill edges defining a minimal triangulation of G.

$E_H \leftarrow E$; $H \leftarrow (V, E_H)$; $F \leftarrow \emptyset$; $V' \leftarrow V$; *prev-card* $\leftarrow 0$; $s \leftarrow 0$; $V_T \leftarrow \emptyset$; $E_T \leftarrow \emptyset$;
for i *from* n *downto* 1 **do**
  Choose a vertex x in V' such that $|N_H(x) - V'|$ is maximum;
  $\alpha(i) \leftarrow x$; *new-card* $\leftarrow |N_H(x) - V'|$ // $= |Madj_H(x)|$;

  if *new-card* $\leq$ *prev-card* //*begin new clique* **then**
    if *new-card* $= 0$ **then**
      $s \leftarrow s + 1$; $V_T \leftarrow V_T + \{s\}$; $Atom(s) \leftarrow \emptyset$;
    else
      $k \leftarrow \min\{j \mid \alpha(j) \in Madj_H(x)\}$; $p \leftarrow ato(\alpha(k))$;
      if $Madj_H(x)$ *is a clique in* G **then**
        $s \leftarrow s + 1$; $V_T \leftarrow V_T + \{s\}$; $Atom(s) \leftarrow Madj_H(x)$;
        $E_T \leftarrow E_T + \{ps\}$; $CliqueMinSep(ps) \leftarrow Madj_H(x)$;
      else
        $s \leftarrow p$ //*merge new clique with clique* p;
  $ato(x) \leftarrow s$: $Atom(s) \leftarrow Atom(s) + \{x\}$;
  foreach $y \in V' - N_H[x]$ **do**
    if *there is an* xy*-path* μ *in* $G(V')$ *such that for each internal node* z *of* μ $|N_H(z) - V'| < |N_H(y) - V'|$ **then**
      $F \leftarrow F + \{xy\}$; $E_H \leftarrow E_H + \{xy\}$;
  $H \leftarrow (V, E_H)$; $V' \leftarrow V' - \{x\}$; *prev-card* $\leftarrow$ *new-card*;

Theorem 4.1 *Let $G = (V, E)$ be a connected graph. Algorithm MCSM-Atom-Tree computes an atom tree of G in $O(nm)$ time.*

Proof: The correctness follows from the correctness of Algorithm Expanded-MCSM and Theorem 3.5. $O(nm)$ time complexity follows from the $O(nm)$ time complexity of Algorithm Expanded-MCSM and from the fact that determining whether $Madj_H(x)$ is a clique in G costs $O(m)$ time, and therefore $O(nm)$ time globally. □

Let us remark that the clique tree of a chordal graph computed by the algorithm from [17] is directed, and that each node can be labeled with the vertices which are not part of the father clique. These new vertices correspond to the transitory moplexes of the moplex ordering induced by α; in order to help visualize the atoms, we thus propose a simplified atom tree, which we call the *moplex tree*, bearing only non-redundant information: the (transitory) moplexes label the nodes and the clique minimal separators label the edges. Figure 3 gives this moplex tree for our example.

Example 4.2 *Figure 3 gives graph G from Figure 1, and a numbering α and the fill edges produced by an execution of Algorithm MCSM-Atom-Tree on G; the corresponding atom tree of G is shown: the minimal separator generators*

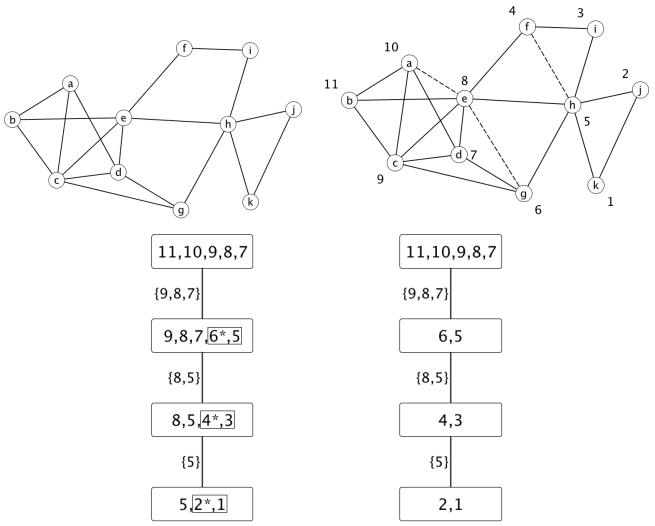

Figure 3: Graph G, the underlying minimal triangulation H of G with the meo computed by MCSM-Atom-Tree, and the corresponding atom tree and moplex tree.

are postfixed by a star; the new vertices (transitory moplexes) in each node label of the atom tree are circled; the moplex tree is also shown.

5 Computing an atom tree from the input graph in $O(n(n + \overline{m}))$ time

In this section, we will make use of the atom tree to devise an algorithm which can compute the clique minimal separator decomposition with a different worst-time complexity than the classical $O(nm)$-time ones discussed in the Introduction: we compute the decomposition in $O(n(n + \overline{m}))$ time.

We first show in Section 5.1 how to traverse the clique tree of a minimal triangulation to ensure $O(n(n+t))$ time, where t is the number of 2-pairs of the minimal triangulation. As discussed above, a 2-pair is a non-edge of the graph, so $t < \overline{m} - f$, where f is the number of fill edges added by the triangulation.

To complement this result, we then go on in Section 5.2 to provide an algorithm which computes a minimal triangulation in $O(n(n + \overline{m}))$ time.

Putting together these two results, we obtain an algorithm which computes the atom tree (and thus the clique minimal separator decomposition) in $O(n(n + \overline{m}))$ time.

5.1 Computing an atom tree from a clique tree in $O(n(n + t))$ time

Once a minimal triangulation has been computed, the bottleneck complexity for computing an atom tree resides in deciding which minimal separators are

cliques in the original graph. To determine this, we will use a new technique. We will consider a directed version of the clique tree (by choosing an arbitrary root). We will use the property that a minimal separator S is a clique in G if and only if it contains no edge of F. We will traverse the clique tree from the root down, and maintain information on the subset X of V represented by the node or edge of the tree we are on. We will maintain for each vertex y of V the number $dF(y)$ of vertices x in the current subset X such that $xy \in F$ (*i.e.* the number of neighbors of y in the graph (V, F) that are in X). Thus X will be a clique in G if and only if for every vertex x of X, $dF(x) = 0$; dF will be updated by vertex additions and removals when going downwards from node u to node v through edge (u, v).

The clique tree is represented by a rooted tree $T = (V_T, Child, Atom)$, where $Child$ maps each node of T to the set of its successors in T, and $Atom$ maps each node of T to a subset of V, which is a maximal clique of H at the beginning, and an atom of G at the end.

Algorithm DF-Atom-Tree

input : A connected graph $G = (V, E)$, the set F of fill edges of a minimal triangulation H of G, a rooted clique tree $T = (V_T, Child, Atom)$ of H, the root r of T.
output: T is merged into the corresponding atom tree of G.
init: $dF_r(x)$ is 0 for each vertex x of V;
foreach $x \in Atom(r)$ **do**
  foreach $y \in V \mid xy \in F$ **do** $dF_r(y) \leftarrow dF_r(y) + 1$;
REC-AT-DF(r, dF_r);
foreach *edge* (u, v) *of* T **do**
  if *not* $Complete(u, v)$ **then** contract u with v ;

REC-AT-DF

input : A node u of T, the mapping dF_u associated with $Atom(u)$
output: The mapping $Complete$ from the set of edges of the subtree of T rooted at u to $\{true, false\}$ defined by: $Complete(v, w) =$ true if and only if $Atom(v) \cap Atom(w)$ contains no edge of F.

foreach $v \in Child(u)$ **do**
  $dF_v \leftarrow dF_u$;
  foreach $x \in Atom(u) - Atom(v)$ **do**
    foreach $y \in V \mid xy \in F$ **do** $dF_v(y) \leftarrow dF_v(y) - 1$;
  $Complete(u, v) \leftarrow true$; // $Atom(u) \cap Atom(v)$ is set as a clique in G;
  foreach $x \in Atom(u) \cap Atom(v)$ **do**
    if $dF_v(x) \neq 0$ **then** $Complete(u, v) \leftarrow false$;
  foreach $x \in Atom(v) - Atom(u)$ **do**
    foreach $y \in V \mid xy \in F$ **do** $dF_v(y) \leftarrow dF_v(y) + 1$;
  REC-AT-DF(v, dF_v);

Example 5.1 *Consider graph H and clique tree T of H from Figure 2: $F = \{ae, eg, fh\}$, represented by dashed edges in H; the root r is node $\{a, c, d, e\}$.*

After the initializing phase of Algorithm DF-Atom-Tree, $dF_r(a) = dF_r(e) =$

$dF_r(g) = 1$ *and* $dF_r(x) = 0$ *for every other vertex of* H.

In the execution of REC-AT-DF(r, dF_r), $Atom(u) = \{a, c, d, e\}$. *When* $Atom(v) = \{a, b, c, e\}$, $Atom(u) - Atom(v) = \{d\}$, *so* $dF_v = dF_r$ *and* $Complete(u, v)$ *is set to false since* $dF_v(a) = dF_v(e) = 1$ ($Atom(u) \cap Atom(v) = \{a, e\}$ *contains edge* ae *of* F). *When* $Atom(v) = \{c, d, e, g\}$, $Atom(u) - Atom(v) = \{a\}$, *so* $dF_v(e)$ *is decremented to* 0 *and* $Complete(u, v)$ *is set to true since* $dF_v(c) = dF_v(d) = dF_v(e) = 0$ ($Atom(u) \cap Atom(v) = \{c, d, e\}$ *contains no edge of* F).

Theorem 5.2 *Let* $G = (V, E)$ *be a connected graph. Given a minimal triangulation* $H = (V, E + F)$ *of* G *and a clique tree* T *of* H, *Algorithm DF-Atom-Tree computes an atom tree of* G *in* $O(n(n+t))$ *time, where* t *is the number of 2-pairs of* H.

To prove this, we will use the following Lemma.

Lemma 5.3 *Let* H *be a connected chordal graph, let* $T = (V_T, Child, Atom)$ *be a rooted clique tree of* H; *then* $\Sigma_{u \in V_T} \Sigma_{v \in Child(u)} |Atom(u) - Atom(v)|$ *is bounded by the number of 2-pairs of* H.

Proof: It is sufficient to show that there is an injective mapping f from the set of triples (x, u, v) such that (u, v) is an edge of T and $x \in Atom(u) - Atom(v)$ to the set of 2-pairs of H. Let $f(x, u, v) = \{x, y\}$, where $y \in Atom(v) - Atom(u)$ (which is not empty since $Atom(u)$ and $Atom(v)$ are maximal cliques of H). Let us show that $\{x, y\}$,is a 2-pair in H, or, equivalently, that $N_H(x) \cap N_H(y)$ is a minimal xy-separator in H. As $Atom(u)$ and $Atom(v)$ are cliques of H containing x and y respectively, $Atom(u) \subseteq N_H(x)$ and $Atom(v) \subseteq N_H(y)$. Hence $Atom(u) \cap Atom(v) \subseteq N_H(x) \cap N_H(y)$. As, by Property 3.6, $Atom(u) \cap Atom(v)$ is an xy-separator in H, $N_H(x) \cap N_H(y)$ is an xy-separator in H too, and it is a minimal one since it is a subset of each xy-separator. Let us show that f is injective. We suppose that $f(x_1, u_1, v_1) = f(x_2, u_2, v_2) = \{x, y\}$, with the root of T_x being at a shorter distance from the root of T than the root of T_y. Then $x_1 = x_2 = x$, $v_1 = v_2 =$ the root of T_y and $u_1 = u_2 =$ the predecessor of v_1 in T. Hence $(x_1, u_1, v_1) = (x_2, u_2, v_2)$. □

Proof: (of Theorem 5.2)
Let (u, v) be an edge of T. Determining $Complete(u, v)$ costs $O(n)$ time, and therefore $O(n^2)$ time for all edges of T. A vertex x of V is added exactly once to the current subset X represented by dF (when processing the predecessor of the root of T_x if it exists and when computing dF_r otherwise). Thus vertex additions to X cost $O(n^2)$ time globally. The number of removals from X, *i.e.* $\Sigma_{(u,v) \text{ edge of } T} |Atom(u) - Atom(v)|$, is bounded by t by Lemma 5.3. Thus vertex removals from X cost $O(nt)$ time globally, which puts the complexity of Algorithm DF-Atom-Tree in $O(n(n + t))$ time. □

The complexity bottleneck of Algorithm DF-Atom-Tree resides in the number of vertex removals from dF. There are some instances where this number is small: this is the case when the number of 2-pairs is in $O(n)$, when the number of leaves of T is bounded, and when the minimal triangulation is an interval graph or a path graph. Note that in the general case, the number of vertex removals is often much smaller than the number of 2-pairs.

5.2 Computing a minimal triangulation in $O(n(n+\overline{m}))$ time

We will complete this paper by providing an algorithmic process to compute a minimal triangulation in $O(n(n+\overline{m}))$ time. This algorithm improves the time complexity of [38], which is $O(\overline{m}(\delta^2+\overline{m}))$, where $\overline{m}$ is the number of edges of $\overline{G}$ and δ is the maximum degree of $\overline{G}$ (which can be in $O(n)$).

Our basic process is the following: a minimal triangulation H of a connected graph G can be computed by repeatedly choosing a moplex M, saturating the neighborhood of M and removing M from the graph, until only a clique remains [6], according to Algorithm Moplex-Triangulation given below:

Algorithm Moplex-Triangulation

input : A non-chordal graph $G=(V,E)$.
output: A set F of fill edges defining a minimal triangulation of G.
$V' \leftarrow V$; $F \leftarrow \emptyset$; $G' \leftarrow G$;
repeat
  $M \leftarrow$ find a moplex of G';
  $F' \leftarrow$ edges necessary to saturate $N_{G'}(M)$;
  $F \leftarrow F+F'$; $V' \leftarrow V'-M$; $G' \leftarrow (G'+F')-M$;
until G' *is a clique*;

To find a moplex, one can use algorithm LexBFS from [43], as explained in [5]. LexBFS computes an ordering of the vertices of a graph in linear time [43]. For any LexBFS ordering $\alpha=(\alpha(1),\ldots,\alpha(n))$ of a non-clique graph G, there is a moplex M of G such that $M=\{\alpha(1),\ldots,\alpha(|M|)\}$ [5]; therefore, a moplex of a non-clique graph can be computed in linear time by computing a LexBFS ordering α and defining M as $\{\alpha(1),\ldots,\alpha(k)\}$, where $k+1$ is the smallest integer i such that $N[\alpha(i)] \neq N[\alpha(1)]$.

A moplex can thus be found in linear time; the running time of Algorithm Moplex-Triangulation is in $O(n(m+f))$, since at each step added fill edges are traversed. However, we will see that we can run it in the complement graph, and that thus at each step fill edges are *removed* from the graph.

A LexBFS ordering of G can be obtained by running a slight variant of LexBFS on $\overline{G}$ in $O(n+\overline{m})$ time [35]. Note that for any vertices x and y of G, $N_G[x]=N_G[y]$ if and only if $N_{\overline{G}}(x)=N_{\overline{G}}(y)$, and for any moplex M of G containing x, $N_G(M)=V-(N_{\overline{G}}(x)+M)$. Therefore, Algorithm Moplex-Triangulation can be implemented by initializing the current graph $G'=(V',E')$ with $\overline{G}$, and repeatedly computing our LexBFS ordering α of G', defining M as $\{\alpha(1),\ldots,\alpha(k)\}$ where $k+1$ is the smallest integer i such that $N_{G'}(\alpha(i)) \neq N_{G'}(\alpha(1))$, defining S as $V'-(N_{G'}(\alpha(1))+M)$, removing all edges in S from G', adding the edges that have just been removed from G' to F and removing M from G' until G' has no edge. Initializing G' with $\overline{G}$ costs $O(n^2)$ time, each step of the loop costs $O(n+\overline{m})$ time, the number of steps is bounded by n and determining whether G' has an edge costs $O(n)$ time. The algorithm thus runs in $O(n(n+\overline{m}))$ time.

6 Conclusion

In this paper, we introduce the notion of atom tree of a graph, which not only yields its atoms and its clique minimal separators, but also gives a compact representation of their organization. We study how to compute an atom tree from a clique tree of a minimal triangulation, thereby providing a different approach from the classical one. Previously, computing the decomposition by clique minimal separators had 3 complexity bottlenecks, roughly corresponding to the 3 steps involved:

1. Compute a minimal triangulation.
2. Perform up to n graph searches.
3. Check which of the minimal separators of the minimal triangulation are cliques in the input graph.

As discussed in the Introduction, specialized minimal triangulation algorithms lower the complexity of Item 1 for some graph classes, so this step can be improved for specialized input. Our results have enabled altogether removing Item 2. For Item 3, we provide an algorithm with an $O(n(n + \overline{m} - f))$ time complexity; however, a clever encoding of the clique tree may yield a better algorithm in the future.

We also present an algorithm based on MCS which computes the minimal triangulation and the atom tree at the same time. Recently, [12] unified results for MCS and LexBFS regarding the generation of the maximal cliques and the minimal separators of a chordal graph. Thus LexBFS and LEX M could be extended in the same fashion as MCS and MCS-M to compute an atom tree; the cousin of LexBFS, LexDFS, introduced by Corneil and Krueger [23], should also work, in view of the recent results by Xua, Lia and Liangb [47].

We leave open the question of computing the atoms efficiently without computing a minimal triangulation of the graph. This can be done in linear time when all the clique minimal separators are of size 1 or 2 [40, 29]; for hole-and-diamond-free graphs, this can be done in $O(n^2)$ time [8]; we do not know whether this could be accomplished as efficiently in the general case. One of the areas of investigation would be to define a non-minimal triangulation which preserves clique minimal separators. However, in view of the results by Kratsch and Spinrad [34], which show that finding a clique minimal separator in a graph is at least as hard as finding a triangle, there is little hope in the general case of improving the time complexity of computing the clique minimal separator decomposition to less than $O(n^\alpha)$ time. Finding a process to compute the decomposition which matches the $O(n^\alpha \log n)$ time for computing a minimal triangulation of Heggernes, Telle and Villanger [28] would be an interesting advance and should be possible.

References

[1] S. R. Arikati and C. P. Rangan. An efficient algorithm for finding a two-pair, and its applications. *Discrete Appl. Math.*, 31(1):71-74,1991.

[2] A. Berry, J.R.S. Blair, J.-P. Bordat, G. Simonet. Graph extremities defined by search algorithms. *Algorithms*, 3(2):100–124, 2010.

[3] A. Berry, J. R. S. Blair, P. Heggernes, B. W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287-298,2004.

[4] A. Berry, J. R. S. Blair, G. Simonet. Preface to Special issue on Minimal Separation and Minimal Triangulation. *Discrete Math.*, 306(3):293,2006.

[5] A. Berry, J. P. Bordat. Separability generalizes Dirac's theorem. *Discrete Appl. Math.*, 84(1-3):43-53,1998.

[6] A. Berry, J. P. Bordat. Moplex elimination orderings. *Electron. Notes Discrete Math.*, 8:6-9, 2001.

[7] A. Berry, J.-P. Bordat, P. Heggernes, G. Simonet and Y. Villanger. A wide-range algorithm for minimal triangulation from an arbitrary ordering. *J. of Algorithms*, 58(1):33-66,2006.

[8] A. Berry, A. Brandstädt, V. Giakoumakis, F. Maffray. The atomic structure of hole- and diamond-free graphs. *Submitted.*

[9] A. Berry, E. Dahlhaus, P. Heggernes, G. Simonet. Sequential and parallel triangulating algorithms for Elimination Game and new insights on Minimum Degree. *Theor. Comput. Sci.*, 409(3):601-616,2008.

[10] A. Berry, P. Heggernes, Y. Villander. A vertex incremental approach for maintaining chordality *Discrete Math.*, 306(3):318-336,2006.

[11] A. Berry, R. Krueger, G. Simonet. Maximal Label Search Algorithms to Compute Perfect and Minimal Elimination Orderings. *SIAM J. Discrete Math.*, 23(1):428-446,2009.

[12] A. Berry and R. Pogorelcnik. A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph. *Inform. Process. Lett.*, 111(11):508-511, 2011.

[13] A. Berry, R. Pogorelcnik, G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197-215, 2010.

[14] A. Berry, A. Sigayret. A peep through the looking glass: Articulation points in lattices. *Proceedings of ICFCA 2012, LNAI* 7278:45-60, 2012.

[15] A. Berry, A. Wagler. Triangulation and clique separator decomposition of claw-free graphs. *Proceedings of WG 2012, LNCS, Springer* 7551:7-21,2012.

[16] J. R. S. Blair, P. Heggernes and J. A. Telle. A practical algorithm for making filled graphs minimal. *Theoret. Comput. Sci.*, 250(1-2):125-141, 2001.

[17] J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. *Graph Theory and Sparse Matrix Computation*, 84(1-3): 1-29, 1993.

[18] H.L. Bodlaender and A.M.C.A. Koster. Safe separators for treewidth. *Discrete Math.*, 306:337-350, 2006.

[19] A. Brandstädt, V. Giakoumakis, F. Maffray. Clique separator decomposition of hole-free and diamond-free graphs and algorithmic consequences. *Discrete Appl. Math.*, 160:471-478, 2012.

[20] A. Brandstädt, C.T. Hoàng. On clique separators, nearly chordal graphs, and the Maximum Weight Stable Set Problem. *Theoret. Comput. Sci.*, 389:295-306, 2007.

[21] A. Brandstädt, V.B. Le, S. Mahfud. New applications of clique separator decomposition for the Maximum Weight Stable Set Problem. *Theoret. Comput. Sci.*, 370:229-239, 2007.

[22] A. Brandstädt, V.B. Le, J.P. Spinrad. Graph Classes: A Survey. *SIAM Monographs on Discrete Math. Appl.*, Vol. 3, 1999.

[23] D. G. Corneil and R. Krueger. A unified view of graph searching. *SIAM J. Discrete Math.*, 22(4):1259-1276,2008.

[24] G. A. Dirac. On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg*, 25:71-76, 1961.

[25] D.R. Fulkerson and O.A. Gross. Incidence matrixes and interval graphs. *Pacific Journal of Math.*, 15:835–855, 1965.

[26] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs. *Academic Press*, 1980.

[27] P. Heggernes, Minimal triangulations of graphs: A survey, *Discrete Mathematics*, 306(3):297-317, 2006.

[28] P. Heggernes, J. A. Telle, Y. Villanger. Computing Minimal Triangulations in Time $O(n^{\alpha}logn) = o(n^{2.376})$. *SIAM J. Discrete Math.*, 19(4):900-913,2005.

[29] J. E. Hopcroft, R. E. Tarjan. Efficient algorithms for graph manipulation [h] (algorithm 447). *Commun. ACM*, 16(6):372–378, 1973.

[30] W.-L. Hsu and T.-H. Ma. Substitution decomposition on chordal graphs and its applications. *SIAM J. Comput.*, 28:1004-1020, 1999.

[31] P.-A. Jachiet, R. Pogorelcnik, A. Berry, P. Lopez, E. Bapteste. MosaicFinder: identification of fused gene families in sequence similarity networks. *Bioinformatics*, 29(7):837-44, 2013.

[32] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret, A. Berry. Clustering gene expression data using graph separators. *In Silico Biology*, 7(4-5):433-452, 2007.

[33] D. Kratsch, J. Spinrad. Minimal fill in $O(n^{2.69})$ time. *Discrete Math.*, 306(3):366-371, 2006.

[34] D. Kratsch, J. Spinrad. Between $O(nm)$ and $O(n^{\alpha})$. *SIAM J. Comput.*, 36(2):310-325,2006.

[35] R. Krueger, G. Simonet, A. Berry. A General Label Search to investigate classical graph search algorithms. *Discrete Appl. Math.*, 159(23):128142, 2011.

[36] H.-G. Leimer. Optimal decomposition by clique separators. *Discrete Math.*, 113:99-123, 1993.

[37] D. Meister. Recognition and computation of minimal triangulations for AT-free claw-free and co-comparability graphs. *Discrete Appl. Math.*, 146(3):193-218, 2005.

[38] M. Mezzini and M. Moscarini. Simple algorithms for minimal triangulation of a graph and backward selection of a decomposable markov network. *Theoret. Comput. Sci.*, 411(7-9):958-966, 2010.

[39] K.G. Olesen and A.L. Madsen. Maximal prime subgraph decomposition of Bayesian networks. *IEEE Trans. on Systems, Man and Cybernetics, Part B: Cybernetics* 32(1):21-31, 2002.

[40] R. Paige, R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.* 16 (6): 973-989, 1987.

[41] A. Parra, P. Scheffler. Characterizations and Algorithmic Applications of Chordal Graph Embeddings. *Discrete Appl. Math.* 79(1-3):171-188, 1997.

[42] D. J. Rose. Triangulated graphs and the elimination process. *J. Math. Anal. Appl.*, 32(3):597-609, 1970.

[43] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266-283, 1976.

[44] R.E. Tarjan. Decomposition by clique separators. *Discrete Math.*, 55:221-232, 1985.

[45] R. E. Tarjan, M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566-579, 1984.

[46] S. Whitesides. An Algorithm for Finding Clique Cut-Sets. *Inf. Process. Lett.* 12(1):31-32, 1981.

[47] S.-J. Xua, X. Lia, R. Liangb. Moplex orderings generated by the LexDFS algorithm. *Discrete Appl. Math.* 161(13-14):2189-2195, 2013.