

Split Decomposition and graph-labelled trees: Characterizations and Fully-Dynamic Algorithms for Totally Decomposable Graphs*

Emeric Gioan

Christophe Paul[†]

CNRS - LIRMM, Université de Montpellier 2, France

October 10, 2008

Abstract

In this paper, we revisit the split decomposition of graphs and give new combinatorial and algorithmic results for the class of totally decomposable graphs, also known as the *distance hereditary graphs*, and for two non-trivial subclasses, namely the *cographs* and the *3-leaf power graphs*. Precisely, we give structural and incremental characterizations, leading to optimal fully-dynamic recognition algorithms for vertex and edge modifications, for each of these classes. These results rely on a new framework to represent the split decomposition, namely the *graph-labelled trees*, which also captures the modular decomposition of graphs and thereby unify these two decompositions techniques. The point of the paper is to use bijections between these graph classes and trees whose nodes are labelled by cliques and stars. Doing so, we are also able to derive an intersection model for distance hereditary graphs, which answers an open problem.

*Work supported by the French research grant ANR-06-BLAN-0148-01 “*Graph Decompositions and Algorithms - GRAAL*”. This paper completes and develops the extended abstract [23].

[†]Research partially conducted while C. Paul was on Sabbatical at School of Computer Science, McGill University, Montréal, Canada

1 Introduction

The *1-join composition* and its complementary operation, the *split decomposition*, range among the classical operations in graph theory. It was introduced by Cunningham and Edmonds [7, 8] in the early 80's and has, since then, been used in various contexts, such as perfect graph theory [27], circle graphs [4], clique-width [12] or rank-width [35]. However little is known on the algorithmic aspects of the split decomposition. The first polynomial time algorithm proposed in [7] had $O(n^3)$ time complexity. It was later improved by Ma and Spinrad [32] who described an $O(n^2)$ time algorithm. So far the faster algorithm is due to Dahlhaus [16] and runs in linear time. Though Dahlhaus' algorithm is optimal, it is known as extremely complicated and is not well understood (see e.g. [10]). This lack of algorithmic results may be explained by the fact that no nice or simple representation of the split decomposition has been proposed so far. Existing work still more or less relies on the elusive recursive construction proposed in the seminal papers [7, 8].

In this paper, we introduce an alternative representation of the split decomposition in terms of *graph-labelled trees* on which a notion of *accessibility* is defined. A *graph-labelled tree* is a tree in which any internal node u is labelled by a graph G_u whose vertices, called *marker vertices*, are in one-to-one correspondence with the tree edges incident to u . A graph-labelled tree is associated with a graph, its *accessibility graph*, whose vertex set is the leaf set of the tree. Two vertices x and y of the accessibility graph are adjacent if and only if for any pair $e = uv$ and $e' = vw$ of tree edges on the path in the tree between the leaves corresponding to x and y , the marker vertices corresponding to e and e' are adjacent in G_v (such a path can be seen as alternating between tree-edges and graph-label edges). We can show that the bipartition of the leaves defined by any internal tree-edge (*i.e.* non-incident to a leaf) corresponds to a split in the accessibility graph. It follows that the split decomposition process [7] can be described in terms of graph-labelled trees, which can essentially be seen as a reformulation of previously used representations of the split decomposition (see [6, 29, 22] or [12] in a logic context or [20] in a graph drawing context). The novelty is to focus the split decomposition study on the notion of accessibility. Surprisingly revisiting the split decomposition under this new approach yields new combinatorial and algorithmic results as well as alternative and simpler proof of previously known results. Though important relationships between the split decomposition and the modular decomposition are known (it is a common thought that the split decomposition generalizes the modular decomposition), no unifying framework have been proposed yet.

We will first show that graph-labelled trees provide such a framework. Next, the paper concentrates on *totally decomposable* graphs (with respect to the split decomposition), also known as the *distance hereditary graphs* [3, 24], and related graph classes. These graphs play an important role in other classical decomposition techniques since they are exactly the graphs of rank-width 1 [35] and range among the elementary graphs of clique-width 3 [9]. Notice also that distance hereditary graphs generalize *cographs* which are the graph totally decomposable for the modular decomposition. The *3-leaf powers*, which form a subfamily of chordal distance hereditary graphs, behaves nicely with respect to the split decomposition.

Results and related work. The results we obtain are all based on bijections between constrained graph-labelled trees and the graph classes. The property of being totally decomposable with respect to the split decomposition translates in the graph-labelled tree settings as restricting the graph-labels to cliques and stars. Each of the three graph classes we consider is in one-to-one

correspondence with trees labelled by cliques or stars, that satisfy some simple conditions on the distribution of star and clique labels on the tree nodes.

The first interesting result obtained from these bijections concerns distance hereditary graphs. The definition of accessibility in graph-labelled trees naturally yields a characterization in terms of an intersection model. Though its existence was known [30], such a model was still not discovered (see [38], or [39] page 309).

For each of the three above mentioned graphs classes, namely distance hereditary graphs, cographs and 3-leaf powers, we provide necessary and sufficient conditions under which adding a vertex x adjacent to a certain neighbourhood S in a given graph G , say a distance hereditary one, yields a graph $G' = G + (x, S)$ which is also distance hereditary. Let us call such conditions an *incremental characterization*. Notice that an incremental characterization does not require any constraint on the ordering of vertex insertions. For example, the well-known *simplicial elimination ordering* of a chordal graph, which consists of iteratively adding a vertex which is adjacent to a clique, does not provide an incremental characterization (indeed, if G is chordal graphs, then S does not have to induce a clique for $G + (x, S)$ to be chordal). The challenge in establishing an incremental characterization is to find local conditions, which we obtain for each class. Incremental characterization of cographs was already known [11] but in the context of modular decomposition.

We also propose *edge-modification characterizations*, that are necessary and sufficient conditions under which for a given graph G belonging to a class of graphs, the graph resulting from the addition (or deletion) of an edge e still belongs to the class. Such a characterization was already known for distance hereditary graphs [41] and for cographs [37]. However the characterization proposed in [41], based on the breadth-first search layering structure of distance hereditary graphs [24], is really complicated and requires long and technical proofs. Again using the graph-labelled tree, we generalize the cograph characterization of [37] to a simple edge-modification characterization of distance hereditary graphs.

We use these characterizations (incremental and edge-modification) to design optimal fully-dynamic recognition algorithms for distance hereditary graphs, cographs and 3-leaf powers. Our algorithms maintain a representation (the split tree) of the input graph and support vertex addition and deletion as well as edge addition and deletion in time $O(d)$ where d is the number of edges involved in the modification. Let us point out that the series of modifications to which the input graph is submitted is not known in advance. To be of interest, such an algorithm should not recompute the recognition test from scratch. In order to ensure locality of the computation, most of the known dynamic graph algorithms are based on decomposition techniques. For example, the SPQR-tree data structure has been introduced in order to dynamically maintain the 3-connected components of a graph which allows on-line planarity testing [18]. Existing literature on this problem includes representation of chordal graphs [28], proper interval graphs [25], cographs [37], directed cographs [13], permutation graphs [14]. The data structures used for the last four graph families are strongly related to the modular decomposition tree [21]. So it was a natural to wonder whether the split decomposition can be used to dynamically represent wider graph families? As already announced, we answer positively this question.

Finally, notice that since distance hereditary graphs, cographs and 3-leaf power graphs are hereditary graph families, our fully dynamic recognition algorithms can be used in the context of static graphs as well. This yields, for each of the three graph classes, linear time recognition algorithms. We point out that in the particular case of distance hereditary graphs, our new algorithm is not only competitive but also way simpler than the previous ones [24, 17, 5] and [33]. Also, our

bijjective representations allow to get easy isomorphism tests for elements of these classes.

Outline of the paper. Section 2 introduces the framework of graph-labelled tree and revisits the main results of split decomposition theory. Links with the classical modular decomposition of graph [31] are also established thereby showing that graph-labelled trees provide a unifying view on this two graph decomposition techniques. Section 3 mainly presents incremental characterizations based on the split decomposition for the families of distance hereditary graphs, cographs and 3-leaf power graphs. Optimal vertex-only fully dynamic algorithms are derived from these characterizations in Section 4. Section 5 study the edge-only modification problems for these three graph classes. Again characterizations and optimal fully-dynamic algorithms are proposed.

2 Graph-labelled trees, split and modular decompositions

The purpose of this section is to introduce the notion of *graph-labelled tree* and to show that the theory of split decomposition [7] as well as the theory of modular decomposition [21] can be stated within this framework. Before that, let us first introduce the basic terminology.

Any graph $G = (V(G), E(G))$ we consider is simple and loopless. For a subset $S \subseteq V(G)$, $G[S]$ is the subgraph of G induced by S . If T is a tree and S a subset of leaves of T , then $T(S)$ is the smallest subtree of T spanning the leaves of S . If x is a vertex of G then $G - x = G[V(G) - \{x\}]$. Similarly if $x \notin V(G)$, $G + (x, S)$ is the graph G augmented by the new vertex x adjacent to $S \subseteq V(G)$. Similarly if x and y are two vertices of G such that $xy \notin E(G)$ (resp. $xy \in E(G)$), then define $G + e = G'(V, E \cup \{e\})$ (resp. $G - e = G'(V, E \setminus \{e\})$) with $e = xy$. We denote $N(x)$ the neighbourhood of a vertex x . The neighborhood of a set $S \subseteq V(G)$ is $N(S) = \{x \notin S \mid \exists y \in S, xy \in E(G)\}$. The *clique* is the complete graph and the *star* is the complete bipartite graph $K_{1,n}$. The universal vertex of the star is called its *centre* and the degree one vertices its *extremities*.

2.1 Graph-labelled trees

Definition 2.1 A graph-labelled tree (T, \mathcal{F}) is a tree T in which any node v of degree k is labelled by a graph $G_v \in \mathcal{F}$ on k vertices, called marker vertices, such that there is a bijection ρ_v from the tree-edges incident to v to the marker vertices of G_v .

Let (T, \mathcal{F}) be a graph-labelled tree and l be a leaf of T . A node or a leaf u different from l is *l-accessible* if for any tree-edges $e = wv$ and $e' = vw'$ on the l, u -path in T , we have $\rho_v(e)\rho_v(e') \in E(G_v)$. By convention, the unique neighbour of the leaf l in T is also *l-accessible*. See Figure 1 for an example.

Definition 2.2 The accessibility graph of a graph-labelled tree (T, \mathcal{F}) is the graph $G(T, \mathcal{F})$ whose vertex set is the leaf set of T , and in which there is an edge between x and y if and only if y is *x-accessible*. In this setting, we say that (T, \mathcal{F}) is a graph-labelled tree of $G(T, \mathcal{F})$.

An example of a graph-labelled tree and its accessibility graph is given on Figure 1. Internal vertices of a tree T will be called *nodes*. The name *vertex* will be save for the accessibility graphs. We often abuse the language and call a leaf of T a vertex of the accessibility graph and vice versa if convenient.

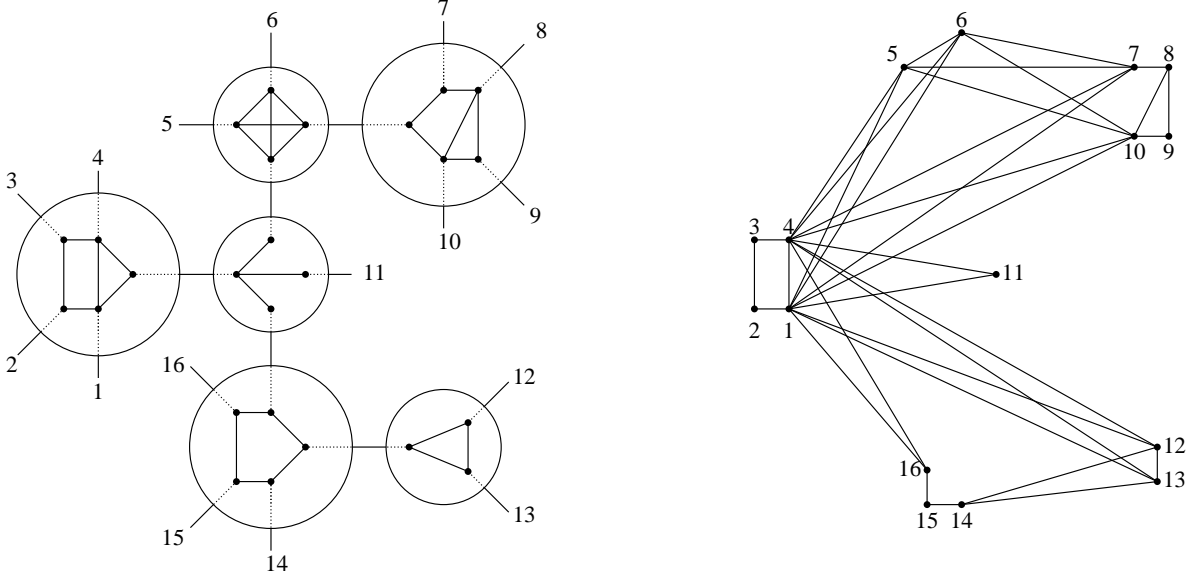


Figure 1: A graph-labelled tree and its accessibility graph. The leaf 12 is 4-accessible (and vice-versa), hence vertices 4 and 12 are adjacent in the accessibility graph. Every node is 4-accessible.

Lemma 2.3 *Let (T, \mathcal{F}) be a graph-labelled tree. The accessibility graph $G(T, \mathcal{F})$ is connected if and only if for any node v of T the graph $G_v \in \mathcal{F}$ is connected.*

Proof: Assume there is a node v of T such that G_v is not connected and let C_v be a connected component of G_v . Let L be the set of leaves belonging to a subtree attached to a marker vertex of C_v . Then by Definition 2.2, none of the leaves of L is l' -accessible for any leaf $l' \notin L$. Thereby in $G(T, \mathcal{F})$, the set of vertices in L is disconnected from the rest of the graph.

Assume for any node v , the graph-label G_v is connected. We prove that $G = G(T, \mathcal{F})$ is connected by induction of the number k of nodes of T . If $k = 1$, this is obviously true since $G(T, \mathcal{F}) = G_v$, where v is the only node of T . Assume that T contains $k > 1$ nodes. Let u be a node such that all its neighbours but one, say v , are leaves (there always exists such a node). Let x_v be the marker vertex of G_v such that $\rho_v(uv) = x_v$. Let (T', \mathcal{F}') be the graph-labelled tree obtained from (T, \mathcal{F}) by replacing u and its leaves by a new leaf l_u . Notice that any leaf l to which x_v is l -accessible is by construction l_u -accessible. Notice that G is obtained from $G' = G(T', \mathcal{F}')$ as follows: replace in $V(G')$ the vertex l_u by the leaves (vertices) attached to u in T ; any new vertex x to which x_v was x -accessible is adjacent to every neighbour of x_v in G' ; the adjacency between the new vertices are those defined by G_u . As by assumption both G' (induction hypothesis) and G_u are connected, G is also connected. \square

From now on, unless explicitly stated, we consider connected graphs (i.e. the graphs belonging to \mathcal{F} in a graph-labelled tree (T, \mathcal{F}) are also connected). The next lemma is central to proofs of further theorems.

Lemma 2.4 *Let (T, \mathcal{F}) be a graph-labelled tree of a connected graph G and let v be a node of T . Then any maximal subtree of $T - v$ contains a leaf l such that v is l -accessible.*

Proof: Let u be a neighbour of node v in T and T_u be the maximal subtree of $T - v$ containing u . The property trivially holds if u is a leaf. So assume T_u contains $k \geq 1$ (non-leaf) node. If u is the only node of T_u , as G_u is connected, there exists a leaf l neighbouring u such that the marker-vertex $\rho_u(lu)$ is adjacent in G_u to the marker-vertex $\rho_u(uv)$. Thereby v is l -accessible. Assume by induction that the property is satisfied for any subtree with $k' < k$ nodes. As G_u is connected, u has a neighbour $w \neq v$ such that $\rho_u(uw)$ and $\rho_u(uv)$ are adjacent in G_u . Let T_w be the maximal subtree of $T_u - u$ containing w . By induction hypothesis, T_w contains a leaf l to which u is l -accessible. by the choice of w , v is also l -accessible. Then it holds for any maximal subtree of $T_u - u$. Let $w \neq v$ be a neighbour of u such that the marker-vertices $\rho_u(wu)$ and $\rho_u(vu)$ are adjacent in G_u . By induction hypothesis the property holds for T_w be the maximal subtree of $T_u - u$ containing w . As G_w is connected, the marker-vertex $\rho_w(wu)$ as a neighbour, say x . The subtree attached to w by the edge $\rho_w^{-1}(x)$ contains a leaf l such that w is l -accessible. By the choice of x , u is also l -accessible. As $\rho_u(wu)$ and $\rho_u(vu)$ are adjacent, v is also l -accessible. \square

Corollary 2.5 *Let (T, \mathcal{F}) be a graph-labelled tree of a connected graph G . Let l be a leaf of T , and $e = uv$, $e' = uv'$ be distinct tree-edges such that u is a l -accessible and e belongs to the u, l path in T . Then $\rho_u(e)\rho_u(e') \in E(G_u)$ if and only if there exists a l -accessible leaf l' in the subtree $T_{v'}$ of $T - e'$ containing v' .*

Proof: If there exists a l -accessible leaf l' in the subtree of $T - e'$ containing v' , then by Definition 2.2, we have $\rho_u(e)\rho_u(e') \in E(G_u)$. So assume $\rho_u(e)\rho_u(e') \in E(G_u)$. By Lemma 2.4, $T_{v'}$ contains a leaf l' such that u is l' -accessible. As u is also l -accessible, then l' is l -accessible. \square

The above Corollary 2.5 can be rephrased as follows: if u and v are two adjacent l -accessible nodes, then there exists a l -accessible leaf l' such that the l, l' -path contains the tree edge uv .

Corollary 2.6 *Let (T, \mathcal{F}) be a graph-labelled tree of a connected graph G . Then any graph $G_v \in \mathcal{F}$ is isomorphic to an induced subgraph of G .*

Proof: Let u_1, \dots, u_k be the neighbours of node v in T and T_1, \dots, T_k be the corresponding subtrees. By Lemma 2.4, for all i , $1 \leq i \leq k$, the subtree T_i contains a leaf l_i such that u is l_i -accessible. It follows that the induced subgraph $G[\{l_1 \dots l_k\}]$ is isomorphic to G_u . \square

Let (T, \mathcal{F}) be a graph-labelled tree of a graph G . Let us observe that a graph-labelled tree of any induced subgraph $H = G[X]$ can be retrieved from (T, \mathcal{F}) . Let $T(X)$ be the smallest subtree of T with set of leaves X . For any $G_v \in \mathcal{F}_X$ labelling a node v of T' , let G'_v be the subgraph induced by the marker-vertices associated with edges belonging to T' . Then set $\mathcal{F}_X = \{G'_v \mid v \in T(X)\}$ and for any $v \in T(X)$, ρ'_v is the bijection between the tree-edges of $T(X)$ incident to v and the vertices of G'_v such that $\rho'_v(e) = x$ if and only if $\rho_v(e) = x$. By construction we have $G(T(X), \mathcal{F}_X) = H$. Let us notice that the degree two nodes of $T(X)$ can be contracted.

2.2 Split decomposition

Definition 2.7 [7] *A split of a graph G is a bipartition (V_1, V_2) of $V(G)$ such that 1) $|V_1| \geq 2$ and $|V_2| \geq 2$; and 2) every vertex of $N(V_1)$ is adjacent to every vertex of $N(V_2)$.*

A graph is *degenerate* (with respect to the split decomposition) if any induced subgraph with at least four vertices has a split. The only degenerate graphs are known to be the cliques and the stars. A graph without any split is called *prime* (with respect to the split decomposition).

The split decomposition of a graph G , as originally studied in [7], consists of: finding a split (V_1, V_2) , decomposing G into $G_1 = G[V_1 \cup \{x_1\}]$, with $x_1 \in N(V_1)$ and $G_2 = G[V_2 \cup \{x_2\}]$ with $x_2 \in N(V_2)$, x_1 and x_2 being called the *marker vertices*; and then recursively decomposing G_1 and G_2 . Though the idea of a tree decomposition appears in [7], Cunningham's main result states the uniqueness of lastly resulting graphs in a split decomposition but does not focus on the structure linking them together. As we will see, the graph-labelled tree framework yields a natural formulation of Cunningham's result in terms of tree.

Lemma 2.8 *Let (T, \mathcal{F}) be a graph-labelled tree and T_1, T_2 be the subtrees of $T - e$ where e is a tree-edge non-incident to a leaf. Then the bipartition (L_1, L_2) of the leaves of T , with L_i being the leaf set of T_i for $i \in \{1, 2\}$, defines a split in the graph $G(T, \mathcal{F})$.*

Proof: Let t_1 and t_2 be the extremities of e and let l_1 and l_2 be leaves of L_1 and L_2 respectively. By definition of $G(T, \mathcal{F})$, l_1 and l_2 are adjacent if and only if t_2 is l_1 -accessible and t_1 is l_2 -accessible. It follows that (L_1, L_2) defines a split of $G(T, \mathcal{F})$. \square

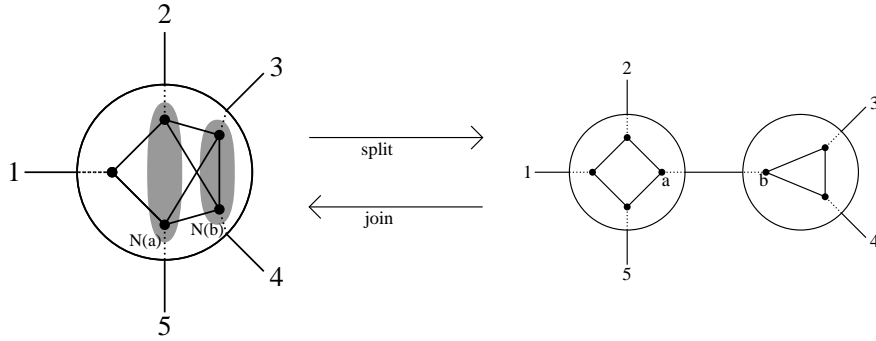


Figure 2: The split and the join operations on a graph-labelled tree.

We can naturally define the *split* operation and its converse, the *join*, on a graph-labelled tree (T, \mathcal{F}) as follows (see Figure 2):

- *Split in (T, \mathcal{F}) :* Let v be a node of T whose graph G_v has a split (A, B) . Let G_A and G_B be the subgraphs resulting from the split (A, B) of G_v and a, b be the respective marker vertices. Splitting the node v consists of substituting v by two adjacent nodes v_A and v_B , respectively labelled by G_A and G_B , such that for any $x \in V(G_A)$ different from a , $\rho_{v_A}^{-1}(x) = \rho_v^{-1}(x)$ and $\rho_{v_A}^{-1}(a) = v_A v_B$ (similarly for any $x \in V(G_B)$ different from b , $\rho_{v_B}^{-1}(x) = \rho_v^{-1}(x)$ and $\rho_{v_B}^{-1}(b) = v_A v_B$).
- *Join in (T, \mathcal{F}) :* Let uv be a tree-edge of T . Then joining the nodes u and v consists of substituting them by a single node w labelled by the graph G_w defined as follows:

$$V(G_w) = (V(G_u) - \{\rho_u(uv)\}) \cup (V(G_v) - \{\rho_v(uv)\})$$

$$E(G_w) = \left((E(G_u) \cup E(G_v)) \cap (V(G_w) \times V(G_w)) \right) \cup \left(N_{G_v}(\rho_v(uv)) \times N_{G_u}(\rho_u(uv)) \right)$$

For any vertex x of G_w , $\rho_w^{-1}(x) = \rho_v^{-1}(x)$ if $x \in V(G_v)$ and $\rho_w^{-1}(x) = \rho_u^{-1}(x)$ if $x \in V(G_u)$.

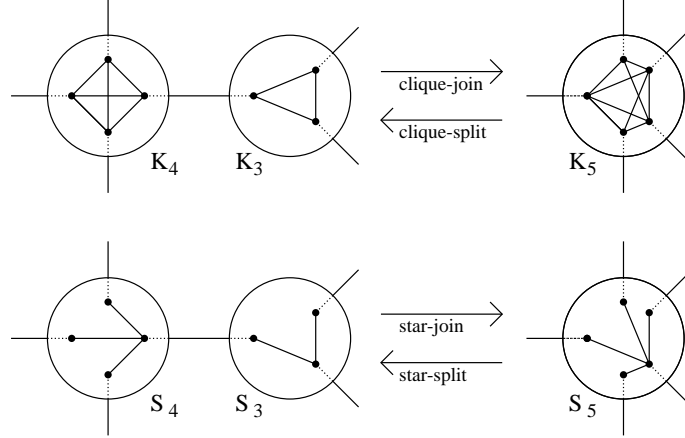


Figure 3: Split and join operations on cliques and stars.

Observe that if (T, \mathcal{F}) is obtained from (T', \mathcal{F}') by a join or a split operation, then it follows from the definitions that $G(T, \mathcal{F}) = G(T', \mathcal{F}')$. These operations show that there is not a unique graph-labelled tree representing a given graph.

Among the join operations, let us distinguish: the *clique-join*, operating on two neighbouring nodes labelled by cliques, and the *star-join*, operating on star-labelled neighboring nodes u, v such that the tree-edge uv links the centre of one star to an extremity of the other. The converse operations are called respectively *clique-split* and *star-split*. See Figure 3. Also, if a node has degree 2 in a graph-labelled tree, then either the accessibility graph is disconnected, or the node can be contracted without loss of information. A graph-labelled tree (T, \mathcal{F}) is *reduced* if every node has degree > 2 and neither clique-join nor star-join can be applied. We are now able to reformulate the main split decomposition theorem first established in [7].

Theorem 2.9 (Cunningham’s Theorem reformulated) *For any connected graph G , there exists a unique reduced graph-labelled tree (T, \mathcal{F}) such that $G = G(T, \mathcal{F})$ and any graph of \mathcal{F} is prime or degenerate.*

For a connected graph G , the *split tree* $ST(G)$ of G is the unique reduced graph-labelled tree (T, \mathcal{F}) in the above Theorem 2.9 (see Figure 1, where the graph-labelled tree is effectively reduced).

Corollary 2.10 *Let $(T, \mathcal{F}) = ST(G)$ be the split tree of a connected graph $G = (V, E)$. Then any split of the graph G is the bipartition induced by removing an edge of (T', \mathcal{F}') , a graph-labelled tree which is obtained from (T, \mathcal{F}) by at most one split operation on a degenerate node.*

The next Lemma will be crucial for algorithm complexity means.

Lemma 2.11 *Let $ST(G) = (T, \mathcal{F})$ be the split tree of a connected graph G . For any vertex $x \in V(G)$, $T(N(x))$ has at most $2 \cdot |N(x)|$ nodes.*

Proof: Let u and v be two adjacent nodes in $T(N(x))$ such that v has degree 2 in $T(N(x))$ and u is on the x, v -path. Let a be the marker vertex of G_v such that $\rho_v^{-1}(a) = uv$. Then a has degree 1 in G_v otherwise, by Corollary 2.5, node v would have degree > 2 . Hence G_v is not prime (the

minimum degree of a prime graph is at least 2), hence it is a star with centre b such that ab is an edge of G_v . Let w be the node neighbor of v such that $\rho_v^{-1}(b) = vw$. If w is not a leaf, then w has degree > 2 in $T(N(x))$, otherwise it would be a star with centre corresponding to vw and the tree would not be reduced. So if $T(N(x))$ is directed from x , any node of degree 2 is followed by a node with degree > 2 . Hence the result. \square

2.3 Modular decomposition

The modular decomposition of a graph is a well understood decomposition process (see [31] for a complete survey). However the purpose of this section is to show that the graph labelled-trees are also a natural tool to represent the modular decomposition. Thereby it provides a framework common to the split and the modular decomposition.

Definition 2.12 *A module of a graph G is a set M of vertices such that any vertex x outside M is either adjacent to all the vertices of M ($M \subseteq N(x)$) or to none of them ($M \cap N(x) = \emptyset$).*

Single vertex sets and the whole vertex set are the *trivial* modules of $G = (V, E)$. A graph is *degenerate* with respect to the modular decomposition, or *M -degenerate* (to avoid confusion with the split decomposition), if any subset of vertices is a module. The *M -degenerate* graphs are cliques or stables (the graph with an empty edge set - or independent set). Intuitively, cliques and *stables* play the same role with respect to the modular decomposition than cliques and stars with respect to the split decomposition. A graph is *prime* with respect to the modular decomposition, or *M -prime*, whenever all its modules are trivial.

If $\mathcal{P} = \{M_1, \dots, M_k\}$ is a partition of the vertex set of a graph G , the *quotient graph* G/\mathcal{P} is defined as the unique (up to isomorphism) subgraph induced by a subset $P \subset V$ such that for all i , $1 \leq i \leq k$, $|P \cap M_i| = 1$. Each vertex $x_i \in P \cap M_i$ is called the *representative* of M_i , for i , $1 \leq i \leq k$.

As the split decomposition, the modular decomposition of a graph $G = (V, E)$ is commonly understood as a recursive process: 1) find a partition of the vertex set V into modules say $\mathcal{P} = \{M_1, \dots, M_k\}$; and 2) recursively decompose the subgraphs $G[M_i]$ for all i , $1 \leq i \leq k$. This naturally yields a rooted tree decomposition. In 1967, Gallai [21] showed that any graph G has a canonical *modular decomposition tree*, denoted $MD(G)$, which is obtained by choosing at the each step of the recursive process the coarsest possible partition. The leaf set of $MD(G)$ is the vertex set of G and each node is labelled by the quotient graph associated with the corresponding partition. These graph labels are either clique, stable or graphs that are *M -prime* graphs. In the usual terminology, clique labelled nodes are called *series* (or 1-nodes) and stable labelled nodes are called *parallel* nodes (or 0-nodes). The canonicity of the modular decomposition tree results from the constraint that no series node (resp. parallel node) is a child of a series node (resp. parallel node). Two vertices x and y are adjacent in G if and only if their representative vertices are adjacent in the quotient graph G/\mathcal{P}_u . Figure 4 shows an example of a graph and its modular decomposition tree.

Let us now describe how the modular decomposition tree $MD(G)$ of a connected graph G naturally transforms into a reduced graph labelled tree (T, \mathcal{F}) whose accessibility graph is G (see Figure 4):

1. Unless the root of $MD(G)$ has degree two, T is isomorphic to the tree underlying $MD(G)$. If $MD(G)$ has a binary root, then T is isomorphic to the tree resulting from the contraction in $MD(G)$ of one of the tree-edges incident to the root.

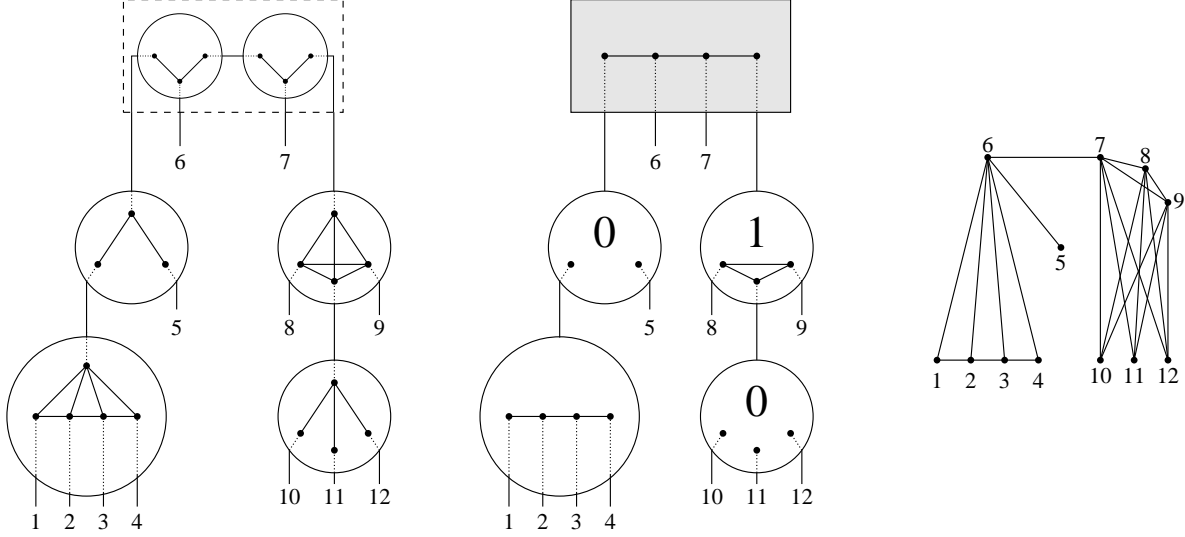


Figure 4: A graph on the right, its modular decomposition tree in the middle, and its split tree on the left. The node in a larger circle is prime for both decompositions. The grey squared node corresponds to the root of the modular decomposition tree. This node is M -prime, but not prime for the split decomposition. Observe that splitting it makes appear stars whose centres are not towards each other. The modular graph-labelled tree is obtained by the converse join operation, i.e. replacing the dashed squared node of the split tree by the grey squared node of the modular decomposition tree.

2. For a node u , distinct from the root of $MD(G)$, with associated quotient graph G/\mathcal{P}_u labelling u in $MD(G)$, the label G_u in (T, \mathcal{F}) is obtained by adding a universal marker vertex to G/\mathcal{P}_u which is mapped to the tree edge uv where v is the father of u in $MD(G)$.

Note that if u is a parallel node in $MD(G)$, then it becomes a star node in (T, \mathcal{F}) . It is straightforward to see from the definitions that G is the accessibility graph of (T, \mathcal{F}) . Let us also point out that the node root u of $MD(G)$ is binary if G has a universal vertex x and $G - x$ is M -prime or if \overline{G} is the disjoint union of two connected components. Finally, (T, \mathcal{F}) is reduced since two series nodes or two parallel nodes are not adjacent in the modular decomposition tree. We will call *modular graph-labelled tree* this graph-labelled tree (T, \mathcal{F}) .

We can now reformulate Gallai's theorem [21] in term of graph-labelled trees.

Theorem 2.13 (Gallai's Theorem reformulated) *For any connected graph G , there exists a unique reduced graph-labelled tree (T, \mathcal{F}) with $G = G(T, \mathcal{F})$ such that T contains a node or an edge r , called the root, and for any node $v \neq r$, we have 1) the graph G_v contains a universal vertex x such that $G_v - x$ is M -prime or M -degenerate, and 2) the edge associated with x in T is on the path between v and r .*

A key property (left as an easy exercise) is that, in $MD(G)$, the label of every non-root node u is M -prime if and only if its corresponding label in (T, \mathcal{F}) is prime for the split decomposition. It follows how simply the split tree $ST(G)$ and the modular graph labelled tree can be retrieved from each other:

- *From the modular graph labelled tree (T_M, \mathcal{F}_M) to the split tree $ST(G)$* : If there is a root node u in T , then substitute the split tree of G_u to node u (i.e. split (T_M, \mathcal{F}_M) according to the splits of G_u and lastly make clique-joins or star-joins to get a reduced graph labelled tree).
- *From the split tree $ST(G)$ to the modular graph labelled tree (T_M, \mathcal{F}_M)* : Test if there exists a node u in $ST(G) = (T, \mathcal{F})$ such that every incident edge but one, say e , is adjacent to a leaf, test if $\rho_u(e)$ is a universal vertex of G_u . If so, then delete u from T (i.e. replace it with a leaf) and repeat until no deletion is possible. The set of remaining nodes induces a subtree T' of T . Then (T_M, \mathcal{F}_M) results from the series of join applied on each internal edge of T' (i.e. substituting a single node labelled by the accessibility graph of T' to T').

It is worth to notice that a subtree of the split tree, namely T' , plays the role of the root of the modular decomposition tree, though, unlike the modular decomposition tree, the split tree is fundamentally unrooted. Figure 4 illustrates these two decompositions on an example.

3 Split-tree characterizations of restricted graph classes

This section presents bijective and incremental characterizations of distance hereditary graphs, cographs and 3-leaf power graphs, in terms of their split tree. The characterization of distance hereditary graphs yields an intersection model which answers an open question (see [39], page 309). Incremental characterizations of each of these three graph classes are also derived. Such a result was already known for cographs [11] (based on the modular decomposition tree), but not for distance hereditary graphs neither for 3-leaf powers. These characterizations will be the basis of the vertex-only fully-dynamic recognition algorithms developed in Section 4.

3.1 Distance hereditary graphs

Definition 3.1 *A graph is distance hereditary (DH for short) if the distance between any given pair of vertices remains the same in any connected induced subgraphs.*

A graph is *totally decomposable* by the split decomposition if any induced subgraph with at least 4 vertices contains a split. By [24], it is known that a graph is DH if and only if it is totally decomposable by the split decomposition, i.e. its split tree is labelled by cliques and stars. Hence DH graphs are exactly accessibility graphs of clique-star labelled trees, *clique-star trees* for short. Among the possible clique-star trees, the split tree is the unique reduced one. In other words, there is a bijection between DH graphs and reduced clique-star trees. Figure 5 gives an example. We mention that ternary clique-star trees were used in [20] to draw DH graphs.

Let us notice that the classical construction of DH graphs [3] (there exist a linear ordering for vertex insertion such that each new vertex y is (a) true twin, (b) false twin, or (c) pendant) is easy to read on the clique-star tree, see Figure 6. We also mention that DH graphs can be characterized by forbidden induced subgraphs [3] (see Section 5 for details).

An intersection model. Given a family S of sets, one can define the intersection graph $\mathcal{I}(S)$ as the graph whose vertices are the elements of S and there is an edge between two elements if and only if they intersect. Many restricted graph families are defined or characterized as the intersection graphs (e.g. chordal graphs, interval graphs... see [30]). Graph families supporting an intersection model can be characterized without even specifying the model [30]. This result applies to DH

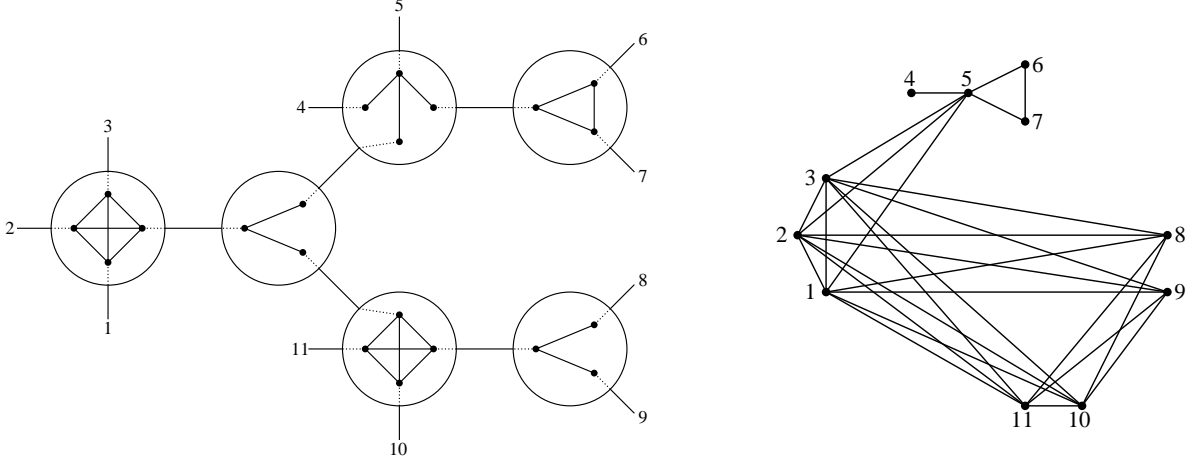


Figure 5: A clique-star reduced tree and its accessibility DH graph.

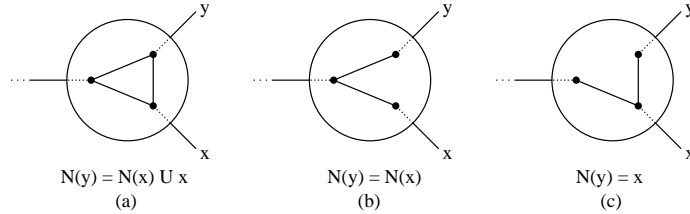


Figure 6: Usual (static) incremental construction of DH graphs: a) adding a true twin y of vertex x consists of inserting a degree 3 clique node on the tree-edge incident to leaf x and attaching leaf y to that node; b) adding a false twin y of vertex x consists of inserting a degree 3 star node on the tree-edge incident to leaf x such that x and y are mapped to the extremities of the star; and c) adding a pendant vertex y to vertex x consists of inserting a degree 3 star node whose centre is mapped to x and to which y is attached.

graphs, and no model was known (see [39], page 309). Based on clique-star trees, an intersection model can be easily derived. Note that it can be equivalently stated by considering only reduced clique-star trees, or even only ternary ones. We call *accessibility set* of a leaf l in a graph-labelled tree the set of pairs $\{l, l'\}$ with l' a l -accessible leaf, or, equivalently, the set of paths in the tree joining l to a l -accessible leaf l' .

Theorem 3.2 (Intersection model) *A graph is distance hereditary if and only if it is the intersection graph of a family of accessibility sets of leaves in a set of clique-star trees.*

Proof: Follows directly from the representation of DH graphs as accessibility graphs of clique-star trees. \square

Incremental characterization. Let G be a connected DH graph and let $ST(G) = (T, \mathcal{F})$ be its split tree. Given a subset S of $V(G)$ and $x \notin V(G)$, we want to know whether the graph $G + (x, S)$ is DH or not.

Definition 3.3 *Let $T(S)$ be the smallest subtree of T with leaves S . Let u be a node of $T(S)$.*

1. u is fully-accessible if any subtree of $T - u$ contains a leaf $l \in S$;
2. u is singly-accessible if it is a star-node and exactly two subtrees of $T - u$ contain a leaf $l \in S$ among which the subtree containing the neighbor v of u such that $\rho_u(uv)$ is the centre of G_u ;
3. u is partially-accessible otherwise.

We say that a star node is *oriented towards* an edge (or a node) of T if the tree-edge mapped to the centre of the star is on the path between that edge (or node) and the star.

Theorem 3.4 (Vertex incremental characterization) *Let G be a connected distance hereditary graph and $ST(G) = (T, \mathcal{F})$ be its split tree. Then $G + (x, S)$ is distance hereditary if and only if:*

1. at most one node of $T(S)$ is partially accessible;
2. any clique node of $T(S)$ is either fully or partially-accessible;
3. if there exists a partially accessible node u , then any star node $v \neq u$ of $T(S)$ is oriented towards u if and only if it is fully accessible; otherwise, there exists a tree-edge e of $T(S)$ towards which any star node of $T(S)$ is oriented if and only if it is fully accessible.

Proof:

\Rightarrow Since $G + (x, S)$ is a DH graph, it is the accessibility graph of a ternary clique-star tree $(\tilde{T}, \tilde{\mathcal{F}})$. Let u be the node of \tilde{T} to which x is attached and let v, w be its neighbours. Now consider the clique-star tree (T', \mathcal{F}') obtained by applying any possible clique-join or a star-join to tree-edges different from uv and uw . Notice that $ST(G)$ is obtained by 1) removing x and u in T' , 2) making vw adjacent, and 3) if needed apply a join on the new tree-edge vw .

Assume the join on vw is not required to obtain $ST(G)$. Then any node of $T(S)$ is a node of T' and as any leaf of S was x -accessible in T' , the three conditions are a consequence of Corollary 2.5. In that case, all the fully accessible star-nodes are oriented towards the tree-edge vw .

Assume $ST(G)$ is obtained after a join on vw which result on a new node u' . Then any node of $T(S)$ but u' of $ST(G)$ corresponds to a node of T' . Again by Corollary 2.5 the nodes of $ST(G)$ different than u' are all singly or fully-accessible. It is straightforward to check that u' is partially accessible and that the third condition holds (as a consequence of Corollary 2.5).

\Leftarrow Assume there is no partially accessible node, so there exists a tree-edge $e = uv$ of T towards which only the fully-accessible star nodes are oriented. Note that e can be chosen such that at least one of its two incident nodes, say u , belongs to $T(S)$. Let (T', \mathcal{F}') be the clique-star tree obtained by: 1) subdividing $e = uv$ into $e_u = uw$ and $e_v = vw$; 2) attaching the leaf x to w (which is thereby a ternary node); 3) making w a clique node if the two subtrees of $T - e$ contain a leaf of S , otherwise w is a star node whose centre is $\rho_w(wu)$.

Any node of $T(S)$ is either fully-accessible or singly-accessible, a degree 2 node in $T(S)$ is singly-accessible. Let w be a node on the tree-path between any $y \in S$ and e and let e_y, e_x be the two tree-edges on that path incident to w . By Definition 3.3, we have that $\rho_w(e_x)\rho_w(e_y)$ is an edge of G_w . It follows that any $y \in S$ is a neighbor of x in $G(T', \mathcal{F}')$. Let us now prove that

any $z \notin S$ is not a neighbor of x in $G(T', \mathcal{F}')$, thereby proving that $G(T', \mathcal{F}') = G + (x, S)$. Let w the node of $T(S)$ which is the closest to the leaf corresponding to z . Notice by the choice of w , w cannot be fully accessible (otherwise it would not be the closest to z). So w is singly-accessible and thereby a star node whose centre is oriented toward the subtree of $T - w$ containing z . It follows that the neighbor w' of w on the path between w and e is not z -accessible. Thus z is not a neighbor of x in $G(T', \mathcal{F}') = G + (x, S)$.

Assume there is a partially accessible node u . Then it suffices to split the node u into two new nodes v and w , such that v is adjacent to the neighbors of u not belonging to $T(S)$ and w to those belonging to $T(S)$. Now star-nodes of $T(S)$ are oriented towards the new tree-edge $e = vw$, and the same arguments than above apply.

□

3.2 A split decomposition characterization of cographs

A cograph is a P_4 -free graph [40] (see Figure 7). This graph family is also known as the graphs totally decomposable by the modular decomposition: i.e. their modular decomposition tree does not contain any M -prime node. It follows a graph G is a cograph if and only if its modular graph-labelled tree only contains clique and star nodes (see Theorem 2.13). By the way, this shows that cographs are DH graphs.

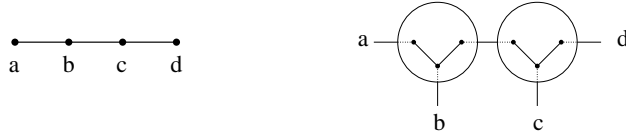


Figure 7: The P_4 is the smallest graph that is not a cograph. Although its split tree only contains star nodes, there is no tree-root toward which the stars are oriented.

Theorem 3.5 (Cograph split tree characterization) *A connected graph $G = (V, E)$ is a cograph if and only if its split tree $ST(G)$ is its modular graph-labelled tree and is a clique-star tree.*

Proof: Assume that G is a cograph. As already mentioned, its modular graph-labelled tree (T, \mathcal{F}) is a clique-star tree. Moreover by definition (T, \mathcal{F}) is reduced, it is also the split tree $ST(G)$.

Assume that G is not a cograph. Then the modular graph-labelled tree contains a node u , labelled by G_u such that G_u is neither a star nor a clique. If G_u is prime with respect to the split decomposition, we are done since $ST(G)$ is not a clique-star tree. So assume the graph G_u contains a split, then the node set of $ST(G)$ and of the modular graph-labelled tree are not the same. That ends the proof. □

Thanks to the construction of the modular graph-labelled tree (see Section 2.3), we can rephrase Theorem 3.5 as follows:

Corollary 3.6 *A connected graph $G = (V, E)$ is a cograph if and only if $ST(G)$ is a clique-star tree and either contains a clique node or a tree edge toward which all the star nodes are oriented. Such a clique-node and tree-edge will be called hereafter the tree-root of $ST(G)$.*

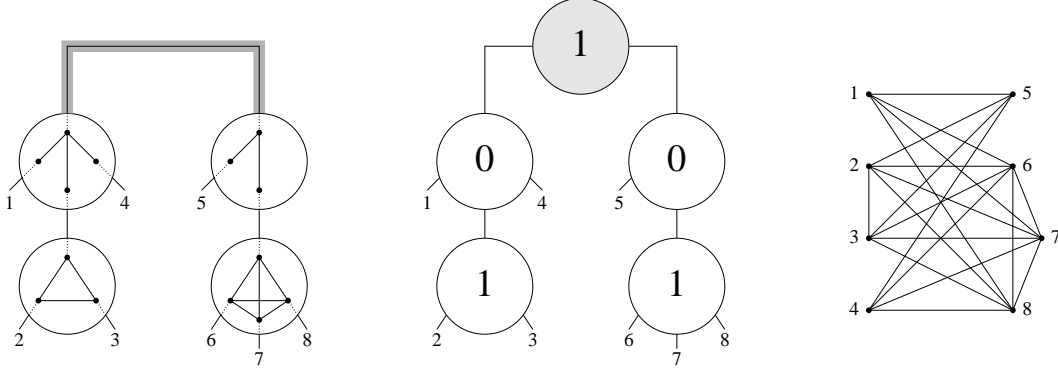


Figure 8: A cograph on the right, its split-tree on the left, and its cotree in the middle. The star nodes, corresponding to 0 labels in the cotree, are oriented towards the tree-root (grey tree-edge).

For the sake of simplicity, let us denote the tree-root of the split-tree $ST(G)$ of a cograph by the set R of nodes of T it contains: that is we set $R = \{u\}$ if the R is a clique-node u and $R = \{u, v\}$ if the R is a tree-edge uv .

Theorem 3.7 (Cograph vertex incremental characterization) *Let G be a connected cograph and $ST(G) = (T, \mathcal{F})$ be its split tree. Then $G + (x, S)$ is a cograph if and only if:*

1. *it is a distance hereditary graph (see conditions of Theorem 3.4) and*
2. *the subtree $T(S)$ of T either contains a node of the tree-root R or contains a node adjacent to a node of R .*

Proof: As in a the split tree of a cograph all the star-nodes are oriented towards the root, $ST(G)$ and thereby $T(S)$ have a natural orientation. This implies that condition 2 above can be rephrased as follows: if $T(S)$ does not contain a node of R , then $T(S)$ has a unique root node which is adjacent to a node of R .

\Rightarrow If $G + (x, S)$ is a cograph, then it is a DH graph. By the structure of their split-tree (see Theorem 3.5), observe that any node of the tree-root is l -accessible for every leaf l . Let us consider the three different ways $ST(G)$ can be transformed into $ST(G + (x, S))$:

1. *Vertex x has been attached to a node u of $ST(G)$. Then the tree-root R of $ST(G)$ is still the tree-root of $ST(G + (x, s))$. So any node of R is x -accessible. Moreover by Corollary 2.5 for at least one node v of R (v may be the same node than u), there are at least two subtrees of $T - v$ not containing u . It follows that R intersects the node set of $T(S)$.*
2. *A node u of $ST(G)$ has been split into two adjacent nodes v and v' and the tree-edge vv' has in turn been subdivided by inserting a new node w adjacent to x . If u does not belong to the tree-root R of $ST(G)$, then as in the first case the tree-root remains unchanged and R intersects the node set of $T(S)$. Assume $R = \{u\}$ (it is a clique-node). It follows from Corollary 2.5 that at least two subtrees of $T - u$ contains a vertex of S as a leaf. Thereby R belongs to the node set of $T(S)$. The tree-root of $ST(G + (x, S))$ is the*

clique-node resulting from the subdivision of u which is x -accessible. So assume that $R = \{u, v\}$, which implies that u is a star-node. Again by Corollary 2.5, $T - u$ contains at least two subtrees of $T - u$ with a leaf in S and at least one of these subtrees is the one containing node v . It follows that R is a subset of the node set of $T(S)$. The tree-root of $ST(G + (x, S))$ contains v and its neighbour resulting from the subdivision of u .

3. A tree-edge of $ST(G)$ has been subdivided by inserting a new node w' adjacent to x . As clique-nodes and star-nodes alternate everywhere in $ST(G)$ but possibly at the tree-root $R = \{u, v\}$, the subdivided tree-edge is: either a) the tree-edge uv joining the vertices of the tree-root; b) or is incident to a leaf l ; or c) incident to unique node u of the tree-root and u is a clique-node. Let us consider these three different cases:
 - (a) Assume the subdivided tree-edge is uv with $R = \{u, v\}$. If the tree-root does not contain a leaf, then node w' is a clique node. It follows that the two subtrees of $T - uv$ contain leaves of S , implying that R is a subset of the node set of $T(S)$. The tree-root of $ST(G + (x, S))$ is now $\{w'\}$. If the tree-root $R = \{u, v\}$ contains a leaf, say v , then $T(S)$ either contains v or S contains two leaves in different subtrees of $T - u$, which implies that the node set of $T(S)$ intersects R . The tree-root of $ST(G + (x, S))$ is either $\{w'\}$ if it is a clique-node or $\{uw'\}$ otherwise.
 - (b) Assume the subdivided edge is wl with l a leaf. Then the tree-root of $ST(G + (x, S))$ is still R and the same arguments than in case 1 above apply.
 - (c) Assume the subdivided tree-edge is uv with $R = \{u\}$ (u is a clique-node). The node w' is a star-node oriented towards the star-node v . In that case at least two subtrees of $T - v$ contains leaves in S and thereby v belongs to $T(S)$. So we are in the situation that $T(S)$ does intersects R but has a neighbor, namely v of the tree-root. Notice also that the tree-root of $ST(G + (x, S))$ is the set $\{w'v\}$.

\Leftarrow We need to show that the second condition implies that all the star nodes are oriented towards the root of $ST(G + (x, S))$ (condition 2 of Theorem 3.5). This is trivially the case if no new node has been created while transforming $ST(G)$ into $ST(G + (x, S))$. This is also true if a new clique-node has been created. So assume that a new star-node w has been inserted. Either the node w arises from the subdivision of a clique-node u or from the subdivision of a tree-edge. Consider the former case. If $\{u\}$ is not the tree-root R of $ST(G)$, then the tree root of $ST(G + (x, S))$ is still R . As nodes of the tree-root are x -accessible, the result follows. Otherwise if $R = \{u\}$, then the new tree root of $ST(G + (x, S))$ is one of the two new clique-nodes resulting from the subdivision of u . The result trivially holds. Consider now the latter case (w is inserted on a tree-edge). This tree-edge has to contain a leaf, say l adjacent to u . If $\{u, l\}$ is not the tree-root R , then as before, R is still the tree-root of $ST(G + (x, S))$ and thereby w is oriented towards R since R is x -accessible. Otherwise if $R = \{u, l\}$, then the tree-root of $ST(G + (x, S))$ is either u (if u is a clique-node) or $\{u, w\}$ (otherwise). It follows that in any cases all the star-nodes of $ST(G + (x, S))$ are oriented towards R .

□

3.3 3-leaf powers

Definition 3.8 A graph $G = (V, E)$ is a k -leaf power if there is a tree T whose leaf set is V and such that $xy \in E$ if and only if the distance in T between leaves x and y is at most k , $d_T(x, y) \leq k$.

The tree T is called *root-tree of G* .

The family of k -leaf power has been introduced in [34] in the context of phylogenetic tree reconstruction. Forbidden induced subgraph characterizations are known for $k \leq 4$. In [2], 3-leaf powers have been characterized as the graphs resulting from substituting cliques into the vertices of a tree. This leads to the following alternative characterization (see Figure 9).

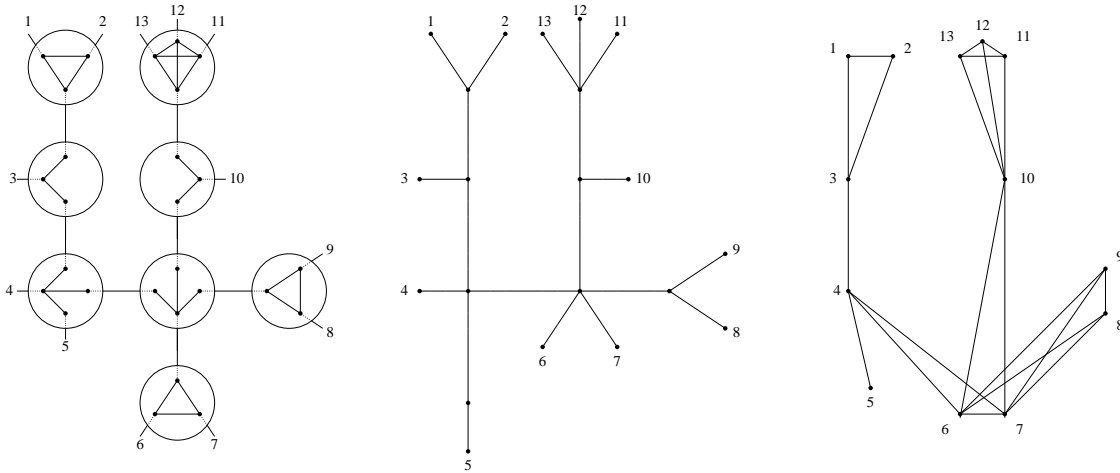


Figure 9: A 3-leaf power graph on the right, its split tree on the left, and a root-tree of this graph in the middle.

Theorem 3.9 (3-leaf power split tree characterization) *A connected graph $G = (V, E)$ is a 3-leaf power if and only if*

1. *its split tree $ST(G) = (T, \mathcal{F})$ is a clique-star tree (G is distance hereditary);*
2. *the set of star nodes forms a connected subtree of T ;*
3. *if u is a star node, then the tree-edge e such that $\rho_u(e)$ is the centre of the star, is incident to a leaf or a clique.*

Proof:

\Rightarrow As G is a 3-leaf power there exists a root-tree T' whose leaf set is V . Assume first that no two leaves are at distance two in T' . We denote by $n(x)$ its unique neighbor in T' of each leaf x . Clearly x and y are adjacent in G if and only if $n(x)$ and $n(y)$ are adjacent in T' . As G is connected, any node of T' is the neighbor of some leaf. We first construct a graph-labelled tree (T', \mathcal{F}') such that $G(T', \mathcal{F}') = G$. The graph label G_v of each internal node $v = n(x)$ of degree $k + 1$ is a star such that $\rho_v(xv)$ is the centre of the star. It is clear that two leaves of (T', \mathcal{F}') are adjacent in $G(T', \mathcal{F}')$ if and only if there are attached to the centre of two neighbouring stars in T' : i.e. $G(T', \mathcal{F}') = G$. Notice that some nodes of T' may have degree 2, then performing a join on the tree-edge not mapped to their centre, provide a new graph-labelled tree (T, \mathcal{F}) which is reduces and which only contains star: this is the split-tree $ST(G)$.

Now assume that T' contains some pairs of leaves at distance 2. Such a pair of leaves defines a pair of true twins in G . The vertex set of G (leaf set of T') can be partitioned into clique modules, each part corresponding to a maximal sets of true twins. Let \mathcal{P} be that partition. The split tree of the quotient graph G/\mathcal{P} is obtained as described above. Now the clique modules are reintroduced by performing true twins insertions (see Figure 6) in the split-tree. Let x be a leaf of $ST(G/\mathcal{P})$ and M_x be the corresponding clique module. Then subdivide the tree-edge incident to x by a clique node of degree $1 + |M_x|$ (see Figure 6 for a true twins augmentation). This yields a split-tree of G having the expected properties.

\Leftarrow Assume that $ST(G) = (T, \mathcal{F})$ satisfies conditions 1 and 2. Then the root-tree T' whose leaf set are V (i.e. the leaf set of T) is obtained as follows: subdivide any tree-edge $e = vl$ of T incident to a leaf l of T such that v is a star node and $\rho_v(e)$ is not the centre of the star. It is easy to check the G is the 3-leaf power of T' .

□

Theorem 3.10 (3-leaf power vertex incremental characterization) *Let G be a connected 3-leaf power and $ST(G) = (T, \mathcal{F})$ be its split tree. Then $G + (x, S)$ is a 3-leaf power if and only if*

1. *it is a distance hereditary graph (see conditions of Theorem 3.4);*
2. *if $S = \{y\}$, then the leaf y in T neighbors a star node u whose centre is $\rho_u(uy)$; otherwise the subtree $T(S)$ does not contain any partially accessible nodes.*

Proof: Thanks to Theorem 3.9, the case where x is adjacent to a unique vertex y is obvious. So let us assume that $|S| > 1$. Thereby $T(S)$ contains at least one non-leaf node of T .

\Rightarrow Remind that the split tree $ST(G)$ is obtained from the split tree $ST(G + (x, S))$ by removing x and if the node u to which x was attached had degree 3, then joining its other two neighbors by a new tree-edge. It follows from conditions of Theorem 3.9 that $T(S)$ cannot contain a partially accessible node.

\Leftarrow By the construction described in the proof of Theorem 3.4, it is clear that if G is a 3-leaf power and conditions 1 and 2 above are satisfied, then the split tree of $G + (x, S)$ is obtained: either by adding a leaf x attached to a clique node or to a star node u such that $\rho_u(ux)$ is not the centre; or by inserting a new ternary clique node. It follows that the resulting clique-star tree also satisfies the conditions of Theorem 3.9.

□

4 Vertex-only fully-dynamic recognition algorithms

The main result presented in this section is an optimal vertex-only fully dynamic algorithm that maintains the split tree representation of a DH graph. For both insertion and deletion queries, the split tree can be updated in time $O(d(x))$, where $d(x)$ is the degree of the vertex to be inserted or deleted. In the case of an insertion, the algorithm can check whether the resulting graph is DH or not. As corollaries, we obtain linear time recognition and isomorphism algorithms for DH graphs. We also give a short overview of how this algorithm can be specialized for the cases of cographs and of 3-leaf powers.

Let us first describe the data-structure we use to implement the split tree of the input graph.

Data-structure. The following data structure is used to encode the clique-star tree $ST(G) = (T, \mathcal{F})$ of the given connected DH graph G :

1. a (rooted) representation of the tree T . The root of T is chosen arbitrarily and is only required for the seek of computational efficiency;
2. as the graphs of \mathcal{F} are cliques or stars, each node of T only needs a *clique-star mark* distinguishing the type of each node, the degree of the node and in the case of a star a *centre mark* to distinguish its centre from the other marker-vertices;

Such a data structure is clearly an $O(n)$ space representation of any DH graph on n vertices.

4.1 Vertex insertion in DH graphs

The insertion algorithm works in three steps. Given a DH graph G represented by its split tree $ST(G)$ and a new vertex x together with a set of vertices S of G : 1) we first compute the subtree $T(S)$; 2) then we check whether the conditions of Theorem 3.4 are satisfied; and finally 3) if the augmented graph $G + x$ turns out to be DH, we update the split tree data-structure (otherwise the algorithm stop).

Computing the smallest subtree spanning a set of leaves. Given a set S of leaves of a tree T , we need to identify the smallest subtree $T(S)$ spanning S , and to store the degrees of its nodes. This problem is easy to solve on rooted trees by a simple bottom-up marking process with complexity $O(|T(S)|)$. So an arbitrary root of T is maintained (see the data-structure description), and we use the following algorithm.

1. Mark each leaf of S . Along the algorithm, a marked node is *active* if it is not the root and its father is not marked.
2. Each active node marks its father if: 1) the root is not marked and there are at least two active vertices, or 2) the root is marked and there is at least one active node.
3. While the root of the subtree T' induced by the marked nodes is a leaf of T' but does not belong to S , then remove this (root) node from T' , let its child be the new root of T' and check again. Otherwise return $T(S) = T'$. -

By Lemma 2.11, if the augmented graph $G + x$ is DH, the size of $T(S)$ (its number of nodes) is at most $2 \cdot |S|$. To prevent a non-linear complexity in the case $G + x$ is not DH, while computing $T(S)$, we need to count the number of marked nodes. More precisely after step 2, the number of marked nodes is at most $2 \cdot |T(S)|$ (since the number of deleted nodes in step 3 cannot exceed the number of marked nodes). Hence if the graph is DH, this number of marked nodes is at most $4 \cdot |S|$. Whenever more than $4 \cdot |S|$ nodes have been marked during step 2, the algorithm stops and claims that the graph $G + x$ is not DH. In any case, it is easy to check that the above algorithm has running time $O(|S|)$. Its correctness is straightforward.

Testing conditions of Theorem 3.4. The first two conditions of Theorem 3.4 are fairly easy to check by following Definition 3.3: a node u is *fully-accessible* if its degrees in $T(S)$ and T are the same; u is *singly-accessible* if it is a star, if it has degree 2 in $T(S)$ and if the u 's neighbor v , such that $\rho_u(uv)$ is the star centre, belongs to $T(S)$; and u is *partially accessible* otherwise (such a node has to be unique if it exists). These tests cost $O(|T(S)|)$.

We can now assume that the first two conditions of Theorem 3.4 are fulfilled. Since the case $|S| = 1$ is trivial, we also assume that $|S| > 1$.

We define *local orientations* on nodes of a tree as the choice, for each node u , of a node $f(u)$ such that either $f(u) = u$ or $f(u)$ is a neighbor of u . Local orientations are *compatible* if 1) $f(u) = u$ implies $f(v) = u$ for every neighbor v of u , and 2) $f(u) = v$ implies $f(w) = u$ for every neighbor $w \neq v$ of u . An easy exercise is to see that if local orientations are compatible then exactly one of the two following properties holds: either there exists a unique node u with $f(u) = u$, in which case u is called *node-root*, or there exists a unique tree-edge uv with $f(u) = v$ and $f(v) = u$, in which case uv is called *edge-root*.

Testing the third condition of Theorem 3.4 consists of building, if possible, compatible local orientations in the subtree $T(S)$:

1. Let u be a leaf of $T(S)$. Then $f(u)$ is the unique neighbor of u .
2. Let u be a star node of $T(S)$. If u is partially-accessible, then $f(u) = u$. If u is singly-accessible, then $f(u)$ is the unique neighbor v of u belonging to $T(S)$ such that $\rho_u(uv)$ is an extremity of the star. If u is fully-accessible, then $f(u)$ is the neighbor v of u such that $\rho_u(uv)$ is the centre of the star.
3. Let u be a clique node of $T(S)$. If u is partially-accessible, then $f(u) = u$. Otherwise, u is fully-accessible and its neighbors are leaves or star nodes. If $f(v) = u$ for every neighbor v of u then $f(u) = u$. If $f(v) = u$ for every neighbor v of u but one, say w , then $f(u) = w$. Otherwise u is an *obstruction*.

The third condition of Theorem 3.4 is satisfied if and only if 1) there is no obstruction and 2) local orientations of $T(S)$ are compatible. This test can be done in time $O(|T(S)|)$ by a search of $T(S)$. Hence the conditions of Theorem 3.4 can be tested in $O(|T(S)|)$ time. Moreover if the test is satisfied, the search of $T(S)$ locates the node-root or the edge-root.

Updating the split-tree. We now assume that graph $G + (x, S)$ is DH (i.e. conditions of Theorem 3.4 are satisfied). So by Theorem 3.4 the split tree has either a unique node-root or a unique edge-root. To update the split tree, we may subdivide an insertion tree-edge into two new tree-edges. But notice that, since we maintain an (artificial) orientation of the tree, this subdivision can be done in $O(1)$. There are three cases to consider (see Figure 11), after an possible single split preprocess (see Figure 10).

0. *Single split preprocess:* If there is a node-root u being partially-accessible, then under of degree conditions on u , a preliminary update of T consisting of a split of the node u is required. Let U , resp. A , be the set of tree-edges incident to u in T , resp. in $T(S)$.
 - (a) If u is a clique node with $|U \setminus A| \geq 2$, then u is split. Two new adjacent clique nodes v and w are created in T . The marker vertices of v (resp. w) correspond to A , resp. $U \setminus A$,

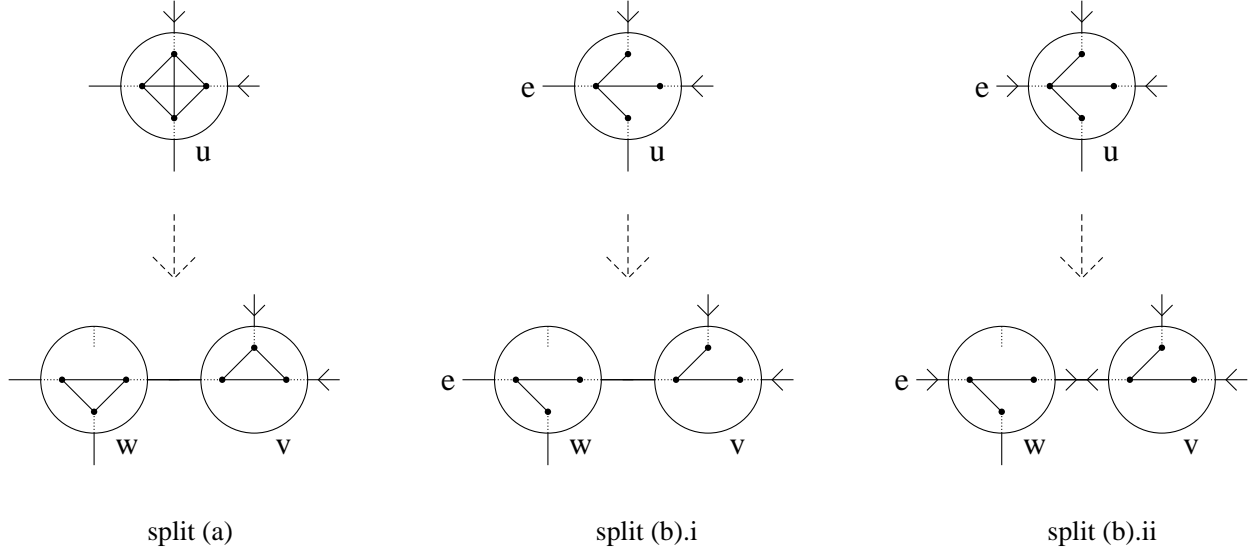


Figure 10: Vertex insertion preprocessing step: a split on the node-root u is required to separate the set A of tree-edges (i.e. those incident to u and belonging to $T(S)$ - drawn with an arrow in the figure) from the others.

except one which corresponds to vw . In this case, v is now the (partially accessible) node-root.

- (b) If u is a star node, the centre of which is mapped to the tree edge e , and $|(U \setminus A) \setminus e| \geq 1$, then u is split and replaced by two adjacent star nodes v and w . Then the extremities of the star G_v correspond to $A \setminus \{e\}$ and its centre to vw (we have $|A \setminus \{e\}| > 1$ since u is not singly accessible), likewise the extremities of the star G_w correspond to $(U \setminus A) \cup \{vw\}$ and its centre to e .
 - i. If $e \notin A$, then the node v becomes the (partially accessible) node-root.
 - ii. If $e \in A$, then the edge vw is now the edge-root.
1. The root of $T(S)$ is a partially accessible node v , or S is reduced to a unique leaf v . Let w be its neighbour in T that does not belong to $T(S)$. Then the insertion edge is $e = vw$, and $ST(G + (x, S))$ is obtained by subdividing vw into two tree-edge vr and rw , where r a degree 3 star node whose centre is $\rho_w(vr)$ and to which x is adjacent. Finally if w is a star with centre $\rho_w(wr)$, we proceed a join operation on the tree-edge wr .
 2. The root of $T(S)$ is a node v which is not partially-accessible. By the definition of the local orientation f , the node v is a clique node, and $ST(G + (x, S))$ is obtained by adding the new leaf x adjacent to v whose degree thereby increases by one.
 3. The root of $T(S)$ is an edge vw . Then $ST(G + (x, S))$ is obtained by subdividing vw with a clique node r of degree 3 and making the leaf x adjacent to r .

Theorem 4.1 (Vertex insertion) *Let $G + (x, S)$ be a graph such that G is a connected distance hereditary graph. Given the data structure of split tree $ST(G)$, testing whether $G + (x, S)$ is distance hereditary and if so computing the data structure of $ST(G + (x, S))$ can be done in $O(|S|)$ time.*

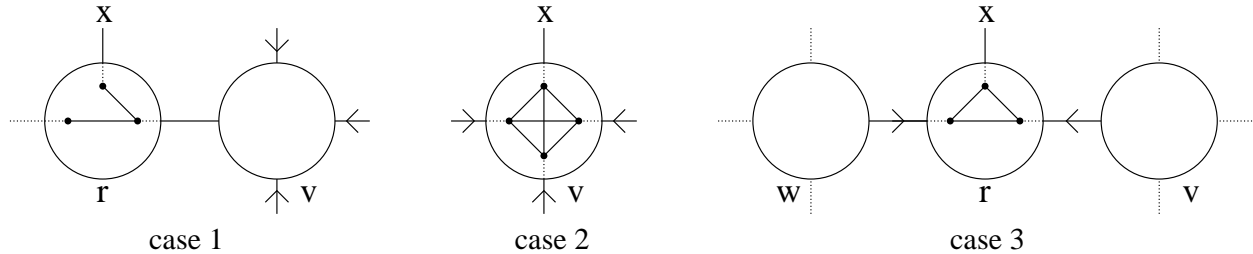


Figure 11: The three different cases for the vertex insertion: 1) the root of $T(S)$ is a partially accessible node v ; 2) the root of $T(S)$ is a node v which is not partially accessible; and 3) the root of $T(S)$ is an edge vw . The figure shows the modified split-tree.

Proof: The correctness follows from the discussion above and the proof of Theorem 3.4.

Concerning the complexity issues, any tree modification operation can be done in $O(1)$ time, except the splitting in case 0 which requires $O(|T(S)|)$ time (by deleting A from u to get w , and adding A to a new empty node v). Any other operation time to maintain the data structure of the split tree (root, degrees...) requires $O(1)$. Then, the complexity for the whole insertion algorithm derives from previous steps and the fact that $O(|T(S)|) = O(|S|)$ (Lemma 2.11). \square

Let us remark that our vertex insertion algorithm yields a linear time recognition algorithm of (static) DH graphs, thereby achieving the best known bound but also simplifying the previous non-incremental ones [24, 17, 5]. It also yields a linear time isomorphism algorithm, thereby achieving the best known bound again with a simpler setting than in [33].

Corollary 4.2 (Static recognition) *The vertex insertion routine enables to recognize distance hereditary graphs in linear time.*

Proof: As the insertion algorithm works only on connected graphs, we have to proceed the vertices in an ordering x_1, \dots, x_n such that, for any $1 \leq i \leq n$, $G[\{x_1, \dots, x_i\}]$ is connected. Any search (e.g. BFS) computes such an ordering in linear time. As the global complexity cost is linear in the sum of the degrees, linear time follows. \square

Corollary 4.3 (Isomorphism) *The vertex insertion routine enables to test distance hereditary graph isomorphism in linear time.*

Proof: To test isomorphism between two DH graphs, it suffices to test isomorphism between the two corresponding split-trees, which can be constructed in linear time by our recognition algorithm and whose size are linear in the number of vertices of the corresponding graphs (Lemma 2.11). Then any linear time tree isomorphism algorithm can be used [1]. \square

4.2 Vertex deletion in DH graphs

Removing a vertex x from a DH graph G always yields a DH graph $G - x$. Let $ST(G)$ be the split tree of G . Updating the data structure of the split tree can be done as follows.

1. Remove the leaf x and update the degree of its neighbor v .

2. If v now has degree 2, then remove v and add an edge between its neighbors u and w . If the resulting clique-star tree is not reduced, proceed a join on the tree-edge uw .
3. If v is a star node whose centre neighbor was x , then $G - x$ is no longer connected, and the split-trees of each connected component are the components of $T - \{v, x\}$.

Lemma 4.4 (Vertex deletion) *Let G be a connected distance hereditary graph and x be a degree d vertex of G . Given the data structure of split tree $ST(G)$, testing whether $G - x$ is a connected distance hereditary graph and if so computing the data structure of $ST(G - x)$ can be done in $O(d)$ time.*

Proof: Any operation, except the join, can be achieved in $O(1)$ time. The complexity of the join on the tree-edge uw is $\min(d(u), d(w))$, where $d(u), d(w)$ are respectively the degree of node u and node w . Since at least one of these nodes is fully accessible, this minimum degree is smaller than d , the degree of x . Hence this join operation costs $O(d)$. \square

To summarize the results of vertex dynamic DH graphs, with Theorem 4.1 and Lemma 4.4, we have proved that:

Theorem 4.5 *There exists a vertex fully dynamic recognition algorithm for connected distance hereditary graphs, maintaining the split tree, with complexity $O(d)$ per vertex insertion or deletion operation involving d edges.*

4.3 Vertex modifications in cographs

To check whether the augmented graph $G + (x, S)$ is a cograph, our vertex insertion algorithm for DH could be used. According to Theorem 3.7, we just need an extra test to verify that the tree-root has a node in the subtree $T(S)$ or is neighbouring a node of $T(S)$. Notice that as the original graph G is a cograph, the star nodes define a natural orientation which can be used to compute $T(S)$. Let us also remark that, as a consequence of Theorem 3.7, the set of singly-accessible nodes (which are stars) has to belong to a path from the tree-root of $ST(G)$ to some node u . It follows that to test condition 3 of Theorem 3.4, the local orientations can be avoided. This path property for the singly-accessible nodes was already noticed (in other terms) in the characterization proposed in [11]. Finally, we need an extra work to update the tree-root as described in the proof of Theorem 3.5. This can also be done in constant time. It follows that the resulting complexity is $O(d)$ by insertion as in the incremental recognition algorithm of Corneil, Perl and Stewart [11] (which is based on the modular decomposition tree).

As cographs are hereditary graphs, the vertex deletion always yields a cograph. Notice also that removing a vertex does not affect the orientation of the remaining star-nodes in the split-tree. It follows that our vertex deletion algorithm for DH graph can be used as well for the vertex deletion of cographs.

Theorem 4.6 *There exists a vertex fully dynamic recognition algorithm for connected cographs, maintaining the split tree, with complexity $O(d)$ per vertex insertion or deletion operation involving d edges.*

4.4 Vertex modifications in 3-leaf powers

Again the DH vertex insertion algorithm can easily be specialized to get work on 3-leaf powers. Thanks to Theorem 3.10, insertion of a pendant vertex x neighbouring y is restricted to the case where a leaf y is attached to the centre of a star node. This can be checked in $O(1)$ time. In the other cases, we just need to test whether the subtree $T(S)$ contains or not a partially accessible node. This only requires a search of $T(S)$ whose size is $O(|S|)$. Concerning the deletion algorithm, as 3-leaf powers are hereditary graphs, we just apply the DH vertex deletion algorithm.

Theorem 4.7 *There exists a vertex fully dynamic recognition algorithm for connected 3-leaf powers, maintaining the split tree, with complexity $O(d)$ per vertex insertion or deletion operation involving d edges.*

Notice that since the family of 3-leaf power is hereditary, this vertex incremental recognition algorithm also applies to static graph. The time complexity is linear as for the recognition algorithm proposed in [2]. Moreover our algorithm can be easily adapted to output the tree root when the input graph is a 3-leaf power.

5 Edge modifications: characterizations and algorithms.

In this section we show that the split tree representation is also the right tool to deal with edge modifications in totally decomposable graphs. Indeed, based on the forbidden induced subgraph characterizations of the three graph families we have considered so far (DH graphs, cographs and 3-leaf powers), we identify necessary and sufficient conditions under which given a graph G and an edge e , the modified graph $G+e$ (or $G-e$) belongs to the same family than G . Using graph-labelled tree representation, these conditions consist in checking if a given path in the split-tree belongs to a small finite set of configurations. This yields to constant time edge fully-dynamic algorithms. Let us mention that such algorithmic results were already known for cographs [37] and DH graphs [41]. For cographs, the edge fully-dynamic algorithm in [37] relies on a modular decomposition based characterization which, again, we are able to transpose in the split decomposition settings, and which are derived as a particular case of the DH edge modification algorithm. Concerning the DH graphs, the constant time algorithm of [41] is way more complicated than the one we propose here. It relies on a tricky analysis on the BFS layering structure [24] of DH graphs and up to our knowledge no simple characterization could be identified from that work. We provide one hereafter. No result of this flavour was known for 3-leaf powers.

5.1 Edge-modification in distance hereditary graphs.

As mentioned above, our edge-modification characterization of DH graphs rely on the forbidden induced subgraph characterization: a graph is distance hereditary if and only if it does not contain a cycle of length at least 5 (C_k for $k \geq 5$), a *gem*, a *house*, nor a *domino* (see figure 12) as induced subgraph [3].

Let G be a connected DH graph and $ST(G) = (T, \mathcal{F})$ be its split tree. If x and y are two vertices of G , we denote $P(x, y)$ the path in T between the leaf x and the leaf y . As $ST(G)$ is a clique-star tree, $P(x, y)$ naturally defines a word $W(x, y)$ whose letters designate the type of the graphs labelling the nodes of $P(x, y)$. A alphabet of four symbols $A = \{K, S, S_x, S_y\}$ is enough to describe $W(x, y)$:

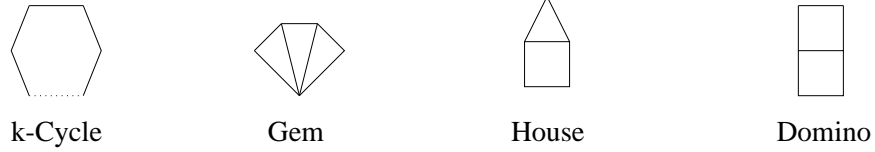


Figure 12: The gem, the house, and the domino are together with the cycles C_k ($k \geq 5$) the forbidden induced subgraphs for DH graphs

- the letter K stands for the clique nodes;
- the letter S stands for the star nodes v , the centre $\rho_v(e)$ of which is mapped to the tree-edge e that does not belong to $P(x, y)$; and
- the letter S_x (resp. S_y) stands for the star nodes v , the centre $\rho_v(e)$ of which is mapped to the tree-edge e that belongs to the subpath of $P(x, y) - v$ containing x (resp. y).

Observe that $xy \in E(G)$ if and only if $W(x, y)$ is S -free (i.e. does not contain the letter S). When describing words, letters in brackets can be deleted: e.g. $K(S)K$ stands for the words KK and KSK .

Theorem 5.1 *Let G be a connected DH graph and $ST(G) = (T, \mathcal{F})$ be its split tree. Let x and y be two vertices of G and $W(x, y)$ be the word labelling the path $P(x, y)$ between x and y in T . Then*

1. *If $xy \notin E$, then $G + xy$ is distance hereditary if and only if $W(x, y)$ is one of the following words:*

$$(S_x)SS(S_y) \qquad (S_x)SK(S_y) \qquad (S_x)KS(S_y) \qquad (S_x)S(S_y)$$

2. *If $xy \notin E$, then $G - xy$ is distance hereditary if and only if $W(x, y)$ is one of the following words:*

$$(S_x)S_yS_x(S_y) \qquad (S_x)S_yK(S_y) \qquad (S_x)KS_x(S_y) \qquad (S_x)K(S_y) \qquad (S_x)(S_y)$$

Moreover if $W(x, y) = (S_x)(S_y)$, then $G - xy$ is no longer connected.

Since the rest of this subsection is devoted to the proof of Theorem 5.1, we first state its algorithmic corollary.

Corollary 5.2 *The following algorithm tests and updates the data-structure of the split tree for the insertion or deletion of an edge xy in a (connected) distance hereditary graph G in constant time.*

1. Test if $W(x, y)$ has length at most 4 and satisfies conditions of Theorem 5.1.
2. Update the split tree of G . Nodes of letters in brackets are called *extreme*.
 - (a) Split any non-extreme node of $W(x, y)$ that is not ternary so that in the resulting clique-star tree, all the non-extreme node of $W(x, y)$ are ternary.
 - (b) Replace the non-extreme nodes by ternary nodes according to the following table. If $W(x, y)$ contains two non-extreme nodes, say u and v , then the neighbor u' of u (resp.

v' of v), that does not belong to $W(x, y)$, becomes adjacent to v (resp. u). See Figure 13. Extreme nodes are left unchanged.

edge insertion \longrightarrow	
\longleftarrow edge deletion	
$(S_x)SS(S_y)$	$(S_x)S_yS_x(S_y)$
$(S_x)SK(S_y)$	$(S_x)S_yK(S_y)$
$(S_x)KS(S_y)$	$(S_x)KS_x(S_y)$
$(S_x)S(S_y)$	$(S_x)K(S_y)$

(c) If necessary, proceed (at most two) join operations involving the nodes that have been changed to get a reduced graph-labelled tree.

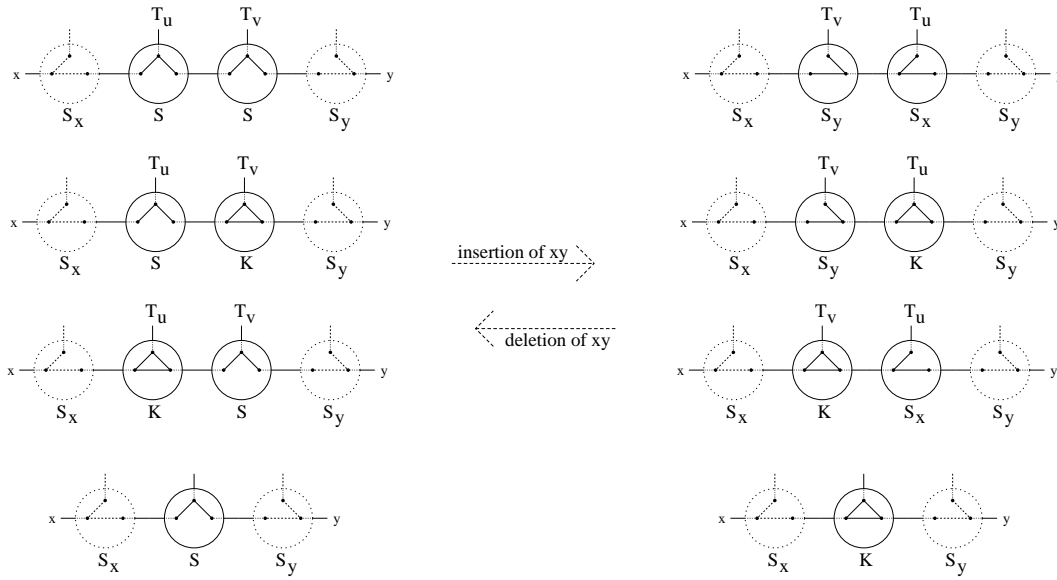


Figure 13: Constant time dynamic algorithm for edge modification in DH graphs

Proof: The correctness of the algorithm is a consequence of Theorem 5.1 and the fact that the split tree transformations are safe (see Figure 13). Let us turn to the complexity analysis. We assume (as we did in Section 4) that an artificial root of the split tree is maintained (remember that the graph and the split tree are connected). Step 1 can be done easily in constant time, by searching the split tree in parallel from x and y towards the root (if the least common ancestor of x and y is found after 4 steps or more, then the length of the path $P(x, y)$ is larger than 4). Step 2 also requires constant time. There are at most two split operations and two join operations respectively at steps (a) and (c), each of which is constant time since it involves a ternary node. And the transformation at step (b) is obviously constant time. \square

In order to prove Theorem 5.1, we first need to introduce some notations and to state some basic properties and technical lemmas. We call *factor*, in a word W , a set of consecutive letters of W . We call *S-maintained subword*, or *subword* for short, a word obtained from W by deleting some letters different from S . As for the clique-star trees, we say that a word $W(x, y)$ is *reduced* if it does

not contain the following factors: KK , S_yS_y , S_xS_x , S_yS and SS_x . Given a word $W = w_1w_2 \dots w_r$ on A , one can associate a clique-star tree P_W which is a path of ternary nodes with hanging leaves (i.e. P_W is a caterpillar). Say that the first and last extreme nodes respectively have leaves x and y , chosen to be the *extremities* of W . Then, the nodes of P_W are labelled by graphs accordingly to the letters of W , w.r.t. x and y as defined previously. We will denote G_W the DH graph obtained as the accessibility graph of the clique-star tree P_W . Let W be a word on A with extremities x, y . We call W *forbidden for edge-insertion* if $xy \notin E(G_W)$ and $G_W + xy$ is not a DH graph; otherwise W is *safe for edge insertion*. Similarly, W is *forbidden for edge-deletion* if $xy \in E(G_W)$ and $G_W - xy$ is not a DH graph; otherwise W is *safe for edge deletion*. The proof of Theorem 5.1 relies on a characterization of the safe words (for insertion and deletion) by forbidden excluded subwords.

Lemma 5.3 *Let x and y be two vertices of a distance hereditary graph G . Then there exists a graph-labelled tree of G with a node u neighbouring leaves x and y such that G_u is isomorphic to $G_{W(x,y)}$. Hence, in particular, it is isomorphic to an induced subgraph of G .*

Proof: By definition, the graph-labelled tree $P_{W(x,y)}$ is isomorphic to the graph-labelled tree obtained from $P(x, y)$ by splitting all nodes in $P(x, y)$ so that the path from x to y is formed by ternary nodes. Joining all these nodes provides a node, which is adjacent leaves x and y and whose label is isomorphic to $G_{W(x,y)}$. It follows from Corollary 2.6 that $G_{W(x,y)}$ is an induced subgraph of G . \square

Lemma 5.4 *Let x and y be two vertices of a distance hereditary graph $G = (V, E)$. If $xy \notin E$, the graph $G + xy$ is distance hereditary if and only if the word $W(x, y)$ is not forbidden for edge-insertion. If $xy \in E$, the graph $G - xy$ is distance hereditary if and only if the word $W(x, y)$ is not forbidden for edge-deletion.*

Proof: By definition, the graph $G_{W(x,y)} + xy$ is DH if and only if $W(x, y)$ is not forbidden for edge insertion. We prove that $G + xy$ is DH if and only if $G_{W(x,y)} + xy$ is DH. By Lemma 5.3, the graph $G_{W(x,y)}$ is isomorphic to the label of a node u with adjacent leaves x and y in a graph-labelled tree (T, \mathcal{F}) of G . Replacing the label of u by $G_{W(x,y)} + xy$ gives a graph-labelled tree of $G + xy$, since the accessibility graph of (T, \mathcal{F}) just changes with the addition of the edge xy , x and y being two leaves of T adjacent to u . Obviously, a graph G is DH if and only if all labels in a graph-labelled tree of G are DH, hence the result. Exactly the same reasoning holds for edge deletion. \square

Lemma 5.5 *Let x and y be two vertices of a distance hereditary graph G . Any connected induced subgraph H of $G_{W(x,y)}$ having x and y as vertices is isomorphic to some graph G_{W_H} where W_H is a (S -maintained) subword of $W(x, y)$, and conversely.*

Proof: Let $W = W(x, y) = w_1 \dots w_r$, and let $\{x, z_1, \dots, z_r, y\}$ be the set of vertices of G_W , such that the ordering z_1, \dots, z_r corresponds to the ordering of leaves encountered from x to y in the caterpillar P_W . An induced subgraph H of G_W having x and y as vertices is of the form $H = G_W[V(H)]$ where $V(H) = \{x, z_{i_1} \dots z_{i_k}, y\}$ with $i_1 < \dots < i_k$. Since P_W is a caterpillar, H is connected if and only if there is an edge between x and a vertex of $V(H) \setminus x$, there is an edge between y and a vertex of $V(H) \setminus y$, and for any j , $1 \leq j \leq k - 1$, there is an edge between a vertex of $\{x, \dots, z_{i_j}\} \subset V(H)$ and a vertex of $\{z_{i_{j+1}}, \dots, y\} \subset V(H)$. By the definition of accessibility, such an edge exists if and only if none of the letters w_j of W such that $z_j \notin V(H)$ is a S . It follows that H is connected if and only if the word $W_H = w_{i_1} w_{i_2} \dots w_{i_k}$ is a (S -maintained) subword of W . Finally, as an edge exists

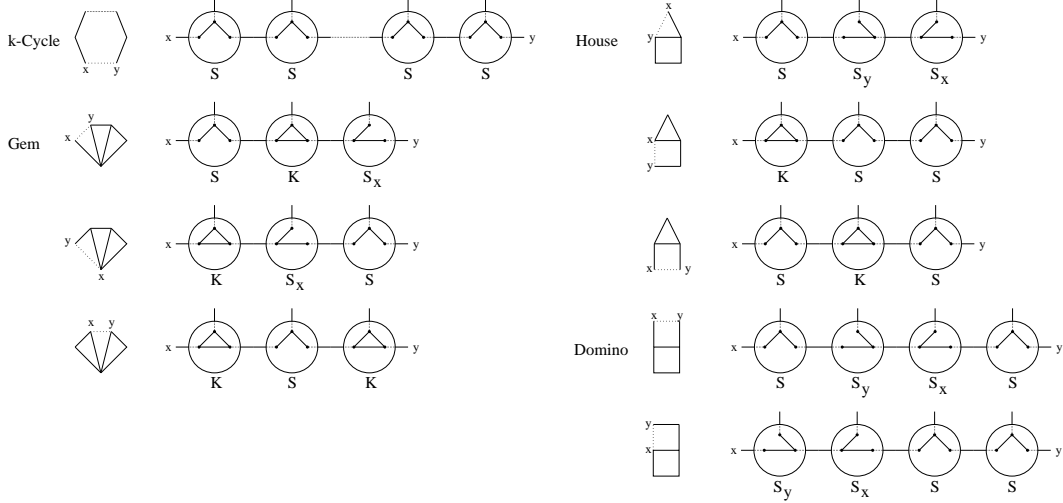


Figure 14: proof of Theorem 5.1, insertion case. Split trees of graphs $H - xy$ for H a DH forbidden induced subgraph. Only useful graphs, i.e. DH ones, are represented. In the table of insertion forbidden subwords, in comparison, repetitions are deleted, and symmetric words are added.

between two vertices of $G_W[V(H)]$ if and only if the corresponding letters in W can be joined by a sequence of letters in $\{K, L, R\}$, we have that $G_W[V(H)]$ is isomorphic to G_{W_H} . Also, the converse is straightforward. \square

Let us consider the DH graphs obtained by removing, resp. adding, an edge xy to one of the DH forbidden induced subgraphs H (cycles, gem, house or domino). It turns out that any of their split tree is a caterpillar with ternary nodes (see Figure 14, resp. Figure 15). Hence, they are determined by their associated words denoted $W_{H-xy}(x, y)$, resp. $W_{H+xy}(x, y)$.

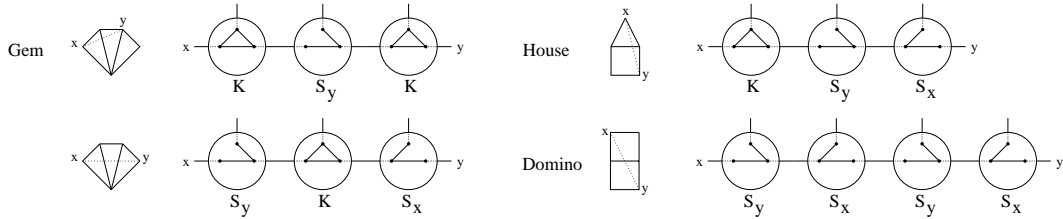


Figure 15: proof of Theorem 5.1, deletion case. Split trees of graphs $H + xy$ for H a DH forbidden induced subgraph. Only useful graphs, i.e. DH ones, are represented. In the table of deletion forbidden subwords, in comparison, repetitions are deleted, and symmetric words are added.

Lemma 5.6 *A word W with extremities x, y is forbidden for edge-insertion, resp. edge-deletion, if and only if it has a subword of type $W_{H-xy}(x, y)$, resp. $W_{H+xy}(x, y)$, for H a distance hereditary forbidden induced subgraph.*

Proof: We prove the statement for edge insertion. Edge deletion case is similar. By definition, a word W , whose extremities are x and y , is forbidden for edge insertion if and only if $G_W + xy$

is not DH, i.e. $G_W + xy$ contains one of the DH forbidden induced subgraphs, say H , which also contains vertices x and y (since G_W is DH). Now by Lemma 5.5, $H - xy$ is a connected induced subgraph of G_W with vertices x and y if and only if $H - xy = G_{W'}$ for some subword W' of W , that is if and only if the word $W' = W_{H-xy}(x, y)$ defined by the caterpillar split tree of $H - xy$ is a subword of W . \square

For each DH forbidden induced subgraph H such that $H - xy$ (resp. $H + xy$) is DH, we obtain a list of (edge-)insertion forbidden subwords, resp. (edge-)deletion forbidden subwords, of type $W_{H-xy}(x, y)$, resp. $W_{H+xy}(x, y)$. They are given by the following tables.

subgraphs	C_k ($k \leq 5$)	Gem	House	Domino
insertion forbidden subwords	$S \dots S$ with $k \geq 3$ S's	SKS_x S_yKS KS_xS SS_yK KSK	SS_yR S_yS_xS KSS SSK SKS	SS_yS_xS S_yS_xSS SSS_yS_x

subgraphs	Gem	House	Domino
deletion forbidden subwords	KS_yK KS_xK S_yKS_x	KS_yS_x S_yS_xK	$S_yS_xS_yS_x$

Proof of Theorem 5.1: By Lemma 5.4 and Lemma 5.6, it remains to show that no subword of $W(x, y)$ belongs to the list of edge-insertion (resp. edge-deletion) forbidden subwords if and only if $W(x, y)$ is one of words described in condition 1, resp. condition 2, of the theorem.

1. Assume that no subword of $W(x, y)$ belongs to the list of forbidden subwords for edge-insertion.

Notice that $W(x, y)$ contains at most two S 's otherwise it would contain a forbidden subwords corresponding to the cycles C_k , $k \geq 5$.

First consider the case $W(x, y)$ contains two S 's. By the KSS , SSK , SKS House's forbidden subwords, $W(x, y)$ has no K letter. By the Domino's forbidden subwords $W(x, y)$ is of the form $(S_x)SS(S_y)$. More precisely, as $W(x, y)$ is reduced, it does not contain the factors S_xS_x or S_yS_y and if a factor with no S contains a S_x (resp. a S_y), then S_x (resp. S_y) has to be the first (resp. last) letter of that factor. It follows that $W(x, y)$ is of the form $(S_x)(S_y)S(S_x)(S_y)S(S_x)(S_y)$. But again since $W(x, y)$ is reduced, it does not contain the factors S_yS or SS_x . Thereby $W(x, y)$ is of the form $(S_x)SS(S_y)$.

Let us now consider the case $W(x, y)$ contains only one S . Hence $W(x, y)$ is of the form wSw' where w and w' are reduced words on $\{K, S_y, S_x\}$. Then, by the SS_yS_x , S_yS_xS House's forbidden subwords, S_yS_x is not a subword of w and w' . By the SS_yK , S_yKS Gem's forbidden subword, KS_x is not a subword of w and w' neither. It follows that w and w' can only be the words $(S_x)(K)(S_y)$. More precisely if w or w' contains a S_x (resp. S_y), then S_x (resp. S_y) has to be the first (resp. last) letter. Moreover by the KSK Gem's forbidden subword, at most one word among w and w' contains a K letter. Finally, since $W(x, y)$ is reduced, it does not contain S_yS or SS_x as factors. Thereby $W(x, y)$ is of the form $(S_x)(K)S(S_y)$ or $(S_x)S(K)(S_y)$.

2. Assume that no subword $W(x, y)$ belongs to the list of forbidden subwords for edge-deletion.

First $W(x, y)$ contains at most one letter K . Otherwise, since it is reduced and contains no factor KK , it would contain a KS_yK or KS_xK Gem's excluded subword.

Assume that $W(x, y)$ contains one letter K . Hence $W(x, y)$ is of the form xKx' where x and x' are reduced words on $\{S_y, S_x\}$. Then, by the KS_yS_x, S_yS_xK House's excluded subwords, x and x' must be of the form $(S_x)(S_y)$. Precisely, if x or x' contains a S_x , resp. S_y , then S_x , resp. S_y , must be the first letter, resp. last. Moreover, by the S_yKS_x Gem's excluded subword, if x contains a S_y , resp. x' contains a S_x , then x' does not contain a S_x letter, resp. x does not contain a S_y . Hence $W(x, y)$ is of the form $(S_x)(S_y)K(S_y)$ or $(S_x)K(S_x)(S_y)$.

Assume $W(x, y)$ does not contain the letter K . Then, by the $S_yS_xS_yS_x$ Domino's excluded subword, $W(x, y)$ must be of the form $(S_x)(S_y)(S_x)(S_y)$, where any letter in brackets can be deleted if it gives a reduced word, that is of the form $(S_x)S_yS_x(S_y)$ or $(S_x)(S_y)$. \square

5.2 Edge-modification in cographs.

As already mentioned, cographs are P_4 -free graphs (see Figure 7). The split tree of a P_4 on vertices $\{x, y, a, b\}$ is formed by two adjacent star nodes, hence it is associated with the word $W = SS, S_xS_y, S_xS$ or SS_y depending of which leaves x and y correspond to. Adapting Theorem 5.1 and Corollary 5.2 leads to a similar characterization and a similar constant time algorithm, equivalent to the one given in [37] in terms of cotrees.

The characterization and algorithm for connected cographs are obtained simply by replacing, in Theorem 5.1 and Corollary 5.2, the list of possible words $W(x, y)$ and respective transformations of the split tree, by the ones given in the following table.

edge insertion \longrightarrow	
\longleftarrow edge deletion	
SK	S_yK
KS	KS_x
S	K

Using the transformation linking the split decomposition to the modular decomposition, one can check that the above words correspond to the cotree configurations identified in [37] that allow edge insertion or deletion in a cograph.

Theorem 5.7 *There exists an edge fully dynamic recognition algorithm for connected cographs, maintaining the split tree, with complexity $O(1)$ per edge insertion or deletion.*

Proof: Let G be a cograph and x, y be two vertices of G . Since a cograph is DH, the necessary conditions of Theorem 5.1 apply to G , that is: $W(x, y)$ belongs to the lists of words provided by Theorem 5.1. Moreover the word $W(x, y)$ cannot contain SS, S_xS, SS_y or S_xS_y as a subword (otherwise, by Lemma 2.4, a P_4 would exist in G , which is a cograph: contradiction). It is straightforward to check that the only words satisfying these properties in the lists given in Theorem 5.1 are those given in the actual theorem, namely:

1. if $xy \notin E$ and $G + xy$ is a cograph, then $W(x, y)$ is either S, SK or KS ;
2. if $xy \in E$ and $G - xy$ is a cograph, then $W(x, y)$ is either K, S_yK or KS_x .

The transformations of the edge-modification algorithm of cograph (see the above table) are special cases of the ones described and analysed in Corollary 5.2. So the time complexity follows. It remains to check that the transformations of the split-tree, described in the above table, do not create a P_4 in the edge modified graph.

- Assume that $W(x, y) = SK$ and that $G + xy$ has an induced P_4 (the case $W(x, y) = KS$ is symmetric). Let $\{a, b, x, y\}$ be the vertices of that P_4 . As the split-tree of G is partitioned into the path $W(x, y)$ and two subtrees respectively attached to the S node (resp. K node) of $P(x, y)$ and disjoint from $P(x, y)$, the vertices a and b cannot be leaves of the same subtree. Let T_a be the subtree containing the leaf a and T_b be the subtree containing the leaf b . Let us note that since $W(x, y) = SK$, the three vertices y, a and b induce a clique and none of its edges is modified in $G + xy$, contradicting the fact that $\{a, b, x, y\}$ induces a P_4 in $G + xy$.
- Assume that $W(x, y) = S_yK$ and that $G - xy$ has an induced P_4 (the case $W(x, y) = KS_x$ is symmetric). As before let $\{a, b, x, y\}$ be the vertices of that P_4 and let T_a (resp. T_b) be the subtree of $ST(G) - W(x, y)$ containing the leaf a (resp. b). This again implies that $\{a, b, y\}$ induces a clique that is not modified by the removal of xy , contradicting the fact that $\{a, b, x, y\}$ induces a P_4 in $G - xy$.
- Assume $W(x, y) = S$ and thus $xy \notin E$ (resp. $W(x, y) = K$ and thus $xy \in E$), then x and y are false (resp. true) twins in G (xa is an edge if and only if ya is an edge). This remains unchanged by the insertion (resp. deletion) of xy : there is no P_4 in $G + xy$ (resp. $G - xy$) containing x and y : $G + xy$ (resp. $G - xy$) is a cograph.

□

5.3 Edge-modification in 3-leaf powers.

As 3-leaf power are DH, the edge insertion must satisfy the properties of the edge insertion in DH graph. As a corollary of Theorem 3.9, the split tree of a 3-leaf power graph does not contain a path of three nodes labelled successively by a star, a clique, and a star. Hence, the words $SKS_x, S_yKS, S_xKS_x, S_yKS_y$ and SKS , have to be deleted from the list of Theorem 5.1. This simplification turns out to be not sufficient, conditions on the degrees and on adjacent nodes have to be added.

The characterization and algorithm for connected 3-leaf power graphs are obtained by:

1. replacing, in Theorem 5.1 and Corollary 5.2, the list of possible words $W(x, y)$ and respective transformations of the split tree, by the ones given in the following table.

edge insertion \longrightarrow	
\longleftarrow edge deletion	
$(S_x)SK$	$(S_x)S_yK$
$KS(S_y)$	$KS_x(S_y)$
$(S_x)S$	$(S_x)K$
$S(S_y)$	$K(S_y)$

2. adding supplementary conditions on the safe words:

- (a) In the cases of words $(S_x)SK, (S_x)S_yK, KS(S_y)$, and $KS_x(S_y)$, the non-extreme letter (i.e. not in brackets) corresponding to a star must come from a ternary star node in

$ST(G)$ and such that the adjacent vertex of the tree not in the path is a clique node or a leaf.

- (b) In the case of words S_xS , and SS_y , the node corresponding to a star with letter S must come from a ternary star node in $ST(G)$ and such that the adjacent vertex of the tree not in the path is a clique node or a leaf.
- (c) In the case of the word K , either the corresponding node is ternary and adjacent to a star of which centre is not adjacent to the clique node, or the corresponding node is not ternary and the split tree of G is reduced to this clique node.

Theorem 5.8 *There exists an edge fully dynamic recognition algorithm for connected 3-leaf power graphs, maintaining the split tree, with complexity $O(1)$ per edge insertion or deletion.*

Proof: We recall that, by Theorem 3.9, the split tree of a DH graph is the split tree of a 3-leaf graph if and only if the set of star nodes form a connected subtree and every centre of a star is adjacent to a clique or a leaf. Since a 3-leaf power graph is DH, the necessary conditions of Theorem 5.1 remain necessary for 3-leaf power graphs, that is: $W(x, y)$ is in the list of words provided by Theorem 5.1. Hence, the word $W(x, y)$ cannot contain a letter K between two letters corresponding to stars S , S_x , or S_y . It is straightforward to check that the only words satisfying these properties in the lists given in Theorem 5.1 are those given in the actual theorem plus the associated words $(S_x)SS(S_y)$ and $(S_x)S_yS_x(S_y)$ (which are obtained from each other by respectively the insertion of xy and the deletion of xy). These two words cannot be considered in the list for 3-leaf power graphs, since, in $(S_x)S_yS_x(S_y)$, two star nodes have their centre adjacent to a star, which would contradict Theorem 3.9. The transformations provided by the actual algorithm are simply particular cases of the ones provided in Corollary 5.2.

So it remains to check that the supplementary conditions on the degrees are necessary and sufficient so that under these transformations the edge modified graph is still a 3-leaf power.

- Assume $W(x, y) = (S_x)SK$ is transformed into $(S_x)S_yK$ under the insertion of xy . Let u be the node of $P(x, y)$ which gives the letter S in the path $W(x, y)$. If u is not ternary, it has to be split into two star nodes u' and v , node u' still belonging to $P(x, y)$ (see step 2.a in algorithm of Corollary 5.2). Then in $ST(G + xy)$, node v is made adjacent to the a clique node of $P(x, y)$ (see step 2.b) in algorithm of Corollary 5.2). This is in contradiction with Theorem 3.9 since that clique node neighbors two star nodes. Thereby the S node u is ternary in $ST(G)$ and let T_a (resp. T_b) be the subtree of $ST(G) - P(x, y)$ attached to u (resp. the K node of $P(x, y)$). It follows that $ST(G + xy)$ satisfies the conditions of Theorem 3.9 since T_a is a clique or a leaf, and the subtree T_b a leaf.
- Assume $W(x, y) = (S_x)S_yK$ is transformed into $(S_x)SK$ under the deletion of xy . Let u be the node of $P(x, y)$ which gives the letter S_y in $W(x, y)$. As in the previous case, u has to be a ternary node. Otherwise, it has to be split into two star nodes u' and v , u' still belonging to $P(x, y)$. Again by the transformation algorithm described in Corollary 5.2, the clique node of $P(x, y)$ in $ST(G - xy)$ is neighbouring two star nodes, contradicting Theorem 3.9. Finally, by Theorem 3.9, the node of $ST(G) - P(x, y)$ adjacent to u is a clique or a leaf and the node of $ST(G) - P(x, y)$ adjacent to the clique node of $P(x, y)$ is a leaf. It follows that $ST(G - xy)$ satisfies the conditions of Theorem 3.9.
- The cases $W(x, y) = KS(S_y)$ and $W(x, y) = KS_x(S_y)$ are symmetric to the previous ones.

- Assume $W(x, y) = S_x S$ is transformed into $S_x K$ under the insertion of xy . The same arguments as above imply that the node giving letter S is ternary (and hence has its centre adjacent to a clique or a leaf), otherwise a clique would appear between two stars while inserting xy , contradicting Theorem 3.9. When the word $S_x K$ is transformed into $S_x S$ under the deletion of xy , $ST(G - xy)$ satisfies the conditions of Theorem 3.9 since the clique node in $P(x, y)$ is adjacent to a leaf.
- Assume $W(x, y) = S$ is transformed into K under the insertion of xy . Let u be the node of $ST(G)$ which gives the letter S in $W(x, y)$ and let v be the neighbor of u such that $\rho_u(uv)$ is the centre of the star G_u . By Theorem 3.9, v is either a clique node or a leaf. If u is a ternary node, then $G + xy$ is a clique and hence a 3-leaf power. Otherwise, u has to be split into two star nodes u' and v (as in the previous cases). In $ST(G + xy)$, the node u' neighbouring the leaves x and y is changed into a clique node. It follows that $ST(G + xy)$ satisfies the conditions of Theorem 3.9.
- Assume $W(x, y) = K$ is transformed into S under the deletion of xy . If the clique node u in $P(x, y)$ is not ternary, then all its neighbors in $ST(G)$ have to be leaves. Assume u neighbors a star node w . As the edge modification algorithm splits u into two clique nodes u' and v and then change u' into a star, the clique node v would neighbor two star nodes in $ST(G - xy)$, contradicting Theorem 3.9. So assume u is ternary, then its third neighbor v distinct from x and y is a leaf or a star. If v is a star and $\rho_v(uv)$ is the centre of the star G_v , then the conditions of Theorem 3.9 are not satisfied. Otherwise, it is clear that $ST(G - xy)$ satisfies the conditions of Theorem 3.9.

Finally, we just have to check that the supplementary conditions can be tested in constant time. The fact that a node of $P(x, y)$ is ternary or not can be tested in constant time. When the node is ternary, the fact that the adjacent node not in $P(x, y)$ is a clique, or a leaf, or a star with centre not adjacent to the clique, is also constant time by checking the type of this adjacent node, and in the last case, by checking the marker of the star recording which is the centre vertex and testing if this vertex is adjacent to the ternary node. In the last case where a clique is not ternary, testing if all other nodes of $ST(G)$ are leaves is done simply by testing if G is a clique. \square

References

- [1] A. Aho, J. Hopcroft and J. Ullman. The design and analysis of . *Addison-Wesley*, 1974.
- [2] A. Brandstädt and V.B. Le. Structure and linear time recognition of 3-leaf powers. *Information Processing Letters*, 98(4):133–138, 2006.
- [3] H.-J. Bandelt and H.M. Mulder. Distance hereditary graphs. *Journal of Combinatorial Theory Series B*, 41:182–208, 1986.
- [4] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7:243–254, 1987.
- [5] A. Bretscher. *LexBFS based recognition algorithms for cographs and related families*. PhD thesis, Department of Computer Science, University of Toronto, 2005.

- [6] S. Cicerone and G. Di Stefano. Graph classes between parity and distance-hereditary graphs. *Discrete Applied Mathematics*, 95:197–216, 1999.
- [7] W.H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3:214–228, 1982.
- [8] W.H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3):734–765, 1980.
- [9] D. Corneil, M. Habib, J.-M. Lanlignel, B. Reed, and U. Rotics. Polynomial time recognition of clique-width 3 graphs. In *Latin American Symposium on Theoretical Informatics (LATIN)*, volume 1776 of *Lecture Notes in Computer Science*, pages 126–134, 2000.
- [10] P. Charbit, M. Habib, V. Limouzy, F. de Montgolfier, M. Raffinot and M. Rao. A note on computing set overlap classes. *Information Processing Letters*, 108(4):186–191, 2008.
- [11] D. Corneil, Y. Perl, and L.K. Stewart. A linear time recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.
- [12] B. Courcelle. The monadic second-order logic of graphs XVI: canonical graph decomposition. *Logical Methods in Computer Science*, 2(2):1–46, 2006.
- [13] C. Crespelle and C. Paul. Fully-dynamic recognition algorithm and certificate for directed cographs. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 3353 of *Lecture Notes in Computer Science*, pages 93–104, 2004.
- [14] C. Crespelle and C. Paul. Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 3787 of *Lecture Notes in Computer Science*, pages 38–48, 2005.
- [15] C. Crespelle and C. Paul. Fully dynamic recognition algorithm and certificate for directed cographs. *Discrete Applied Mathematics*, 154:1722–1741, 2006.
- [16] E. Dahlhaus. Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36(2):205–240, 2000.
- [17] G. Damiand, M. Habib, and C. Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263:99–111, 2001.
- [18] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996.
- [19] D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. In Mikhail J. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 8. CRC Press, 1999.
- [20] D. Eppstein, M.T. Goodrich, and J. Yu Meng. Delta-confluent drawings. In *International Symposium on Graph Drawing (GD)*, volume 3843 of *Lecture Notes in Computer Science*, pages 165–176, 2005.
- [21] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Acad. Sci. Hungar.*, 18:25–66, 1967.

- [22] C. Gaviole and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273:115–130, 2007.
- [23] E. Gioan and C. Paul. Dynamic distance hereditary graphs using split decomposition. In *International Symposium on Algorithms and Computation (ISAAC)*, number 4835 in Lecture Notes in Computer Science, pages 41–51, 2007.
- [24] P. Hammer and F. Maffray. Completely separable graphs. *Discrete Applied Mathematics*, 27:85–99, 1990.
- [25] P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM Journal on Computing*, 31(1):289–305, 2002.
- [26] S.-Y. Hsieh, C.-W. Ho, T.-S. Hsu, and M.-T. Ko. Efficient algorithms for the hamiltonian problem on distance-hereditary graphs. In *Annual International Computing and Combinatorics Conference (COCOON)*, volume 2387 of *Lecture Notes in Computer Science*, pages 77–86, 2002.
- [27] W.-L. Hsu. Decomposition of perfect graphs. *Journal of Combinatorial Theory Series B*, 43:70–94, 1987.
- [28] L. Ibarra. Fully dynamic algorithms for chordal graphs. In *Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 923–924, 1999.
- [29] J.-M. Lanlignel. Autour de la décomposition en coupes. PhD Thesis, Université de Montpellier 2 (France), 2001.
- [30] T. McKee and F.R. McMorris. Topics in intersection graph theory. Number 2 in *SIAM Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics (SIAM), 1999.
- [31] R.H. Möhring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.
- [32] T.-H. Ma and J. Spinrad. An $O(n^2)$ algorithm for undirected split decomposition *Journal of Algorithms*, 16:145–160, 1994.
- [33] S.I. Nakano, R. Uehara, and T. Uno. A new approach to graph recognition and application to distance hereditary graphs. In *International Conference on Theory and Applications of Models of Computation (TAMC)*, volume 4484 of *Lecture Notes in Computer Science*, pages 115–127, 2007.
- [34] N. Nishimura, P. Ragde, and D. Thilikos. On graph powers for leaf-labeled trees. *Journal of Algorithms*, 42(1):69–108, 2002.
- [35] S.I. Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory Series B*, 95:79–100, 2005.
- [36] M. Rao. Solving some NP-complete problems using split decomposition. *Discrete Applied Mathematics*, 156(14):2768–2780, 2008.

- [37] R. Shamir and R. Sharan. A fully dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Applied Mathematics*, 136(2-3):329–340, 2004.
- [38] J. Spinrad. List of open problems: www.vuse.vanderbilt.edu/~spin/open.html.
- [39] J. Spinrad. *Efficient graph representation*, volume 19 of *Fields Institute Monographs*. American Mathematical Society, 2003.
- [40] D.P. Sumner. Graphs indecomposable with respect to the x -join. *Discrete Mathematics*, 6:281–298, 1973.
- [41] M. Tedder and D. Corneil. An optimal, edges-only fully dynamic algorithm for distance-hereditary graphs. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *Lecture Notes in Computer Science*, pages 344–355, 2007.
- [42] R. Uehara and Y. Uno. Canonical tree representation of distance hereditary graphs and its applications. Technical Report COMP2005-61, IEICE Technical Report, 2006.

A Appendix

For self-containment of the paper, we provide below a direct proof of Theorem 2.9 in the setting of graph-labelled trees. It relies on next Proposition A.1, somehow the converse of Lemma 2.8, providing also a proof to the well-known fact that the splits of the graph form a bipartitive family.

Proposition A.1 *Let (T, \mathcal{F}) be a reduced graph-labelled tree with prime and degenerate labels obtained by split decomposition of a connected graph $G = (V, E)$. Then any split of the graph G is the bipartition induced by removing an edge of T' , where T' is obtained from (T, \mathcal{F}) by at most one split of a degenerate node.*

Proof: Let (A, B) be a split of G , with $V = A \uplus B$, $A' \subseteq A$, $B' \subseteq B$, and all edges between A and B having their extremities in A' and B' . We consider V as the set of leaves of T . For a node N of T and a vertex v of the label $G_N \in \mathcal{F}$ of N , we say that v is A' -accessible, resp. B' -accessible, if there exists $u \in A'$, resp. $u \in B'$, such that N is u -accessible and v is the marker vertex of G_N associated with the edge of T in path from N to u .

First, we prove the following assertion: if N is a node of T with label G_N and three vertices u, v, w of G_N are such that u is both A' -accessible and B' -accessible, $v \neq u$ is A' -accessible and $w \neq u$ is B' -accessible, then N is a clique and every vertex of G_N is either A' -accessible or B' -accessible. We can assume $v \neq w$. Indeed, if $v = w$, then let x be a vertex of G_N adjacent to u or $v = w$ (it exists by connectivity Lemma 2.3). By Lemma 2.4, there exists a leaf x' of T such that N is x' -accessible and x is associated with the edge of T in the path between N and x' in T . Then x' is adjacent to a vertex in A' and a vertex in B' , hence it belongs to $A' \cup B'$, hence x is A' -accessible or B' -accessible. So we can change v or w into x , and we assume now that $v \neq w$ in the above hypothesis. Since $V = A \uplus B$ is a split of G , there is an edge in G_N between any A' -accessible vertex of G_N and any B' -accessible vertex of G_N . Assume there exist a vertex y of G_N which is not A' -accessible nor B' -accessible. By Lemma 2.4, there exists a leaf z of T such that N is z -accessible and y is associated with the edge of T in the path between N and z in T . Such a leaf z belongs either to $A \setminus A'$ or to $B \setminus B'$, otherwise there would be an edge between y and u in G_N and z would

be adjacent to a vertex in A' and to a vertex in B' , hence it would belong to $A' \cup B'$ and y would be A' -accessible or B' -accessible. Assume for example that z belongs to $A \setminus A'$. Then the vertex y cannot be adjacent to a B' -accessible vertex in G_N . Then we consider the bipartition of vertices of G_N into 1) all A' -accessible vertices except u , plus the vertex y , plus all other vertices which are not A' -accessible nor B' -accessible and not adjacent to a B' -accessible vertex in G_N , and 2) all B' -accessible vertices including u , plus other vertices which are not A' -accessible nor B' -accessible and not adjacent to a A' -accessible vertex in G_N . The two parts of this bipartition have at least two elements, and thus it forms a split of G_N . This implies G_N is degenerate, hence it is a clique since it contains a K_3 . And this implies that a vertex y which is not A' -accessible nor B' -accessible does not exist.

Now consider the subtrees $T[A']$ and $T[B']$ of T spanned respectively by A' and B' . We prove the assertion: $T[A']$ and $T[B']$ have at most one common node. Assume that these two subtrees have at least two common nodes of T . Then they have a common path of T , and then they have a common edge e and two adjacent common nodes R and S , which are the extremities of e . For each of these nodes, by definition of $T[A']$ and $T[B']$, there exists a leaf in A' and a leaf in B' such that the path from the node to the leaf does not contain e . Since all leaves in A' are accessible from all leaves in B' , each node R and S has an A' -accessible vertex and a B' -accessible vertex, not associated with e . That is: both R and S satisfy the hypothesis of the previous assertion. Hence both R and S are cliques, which is a contradiction with the fact that the graph-labelled tree is reduced.

So, there are two possible cases: either 1) there exists an edge e of T such that all leaves in A' are in one connected component of $T - e$, and all leaves in B' are in the other connected component, or 2) there exists a node N such that the connected components of $T - N$ contain either leaves in A' or leaves in B' but not both, and at least two connected components contain leaves in A' resp. B' (otherwise an edge would satisfy the case 1 property).

In the first case, we denote N_A , resp. N_B the extremity of the edge e on the side of A' , resp. B' , vertices. If the bipartition of V induced by $T - e$ is $V = A \uplus B$, then we have the result. Otherwise, there exists for example a leaf a in A in the connected component of $T - e$ containing leaves in B' . Consider the accessibility graph H of this connected component, where the marker vertex associated with e in N_B is a vertex v_e . Since H does not contain vertices in A' , a vertex in A is adjacent in H either to v_e or to another vertex belonging to A . Hence v_e is an articulation vertex of H , such that each connected component of $H - v_e$ has its set of vertices included in $A \setminus A'$ or in B . Then the vertex v_e of N_B is an articulation vertex of its label. Indeed, say that a vertex $v \neq v_e$ is A -accessible, resp. B -accessible, if there exists a leaf $w \in A$, resp. $w \in B$, such that N_B is w -accessible and v is associated to the path between N_B and w . Then the vertices of $N_B - v_e$ are either A -accessible or B -accessible, maybe both, but these two sets are not empty and a A -accessible vertex is not adjacent to a B -accessible vertex. Since the label of N_B has an articulation vertex, it has obviously a split, hence it is degenerate, and it is not a clique, hence it is a star. Then a vertex v of $N_B - v_e$ is either A -accessible or B -accessible, but not both. Otherwise the node at the other extremity of the edge associated with v would have the same property as N_B in terms of articulation vertex, hence it would be a star also, which would be a contradiction with the fact that the graph-labelled tree is reduced. Now, there cannot be a leaf b in B in the connected component of $T - e$ containing leaves in A' , otherwise N_A would also be a star and there would be an edge between a vertex of $A \setminus A'$ and a vertex of $B \setminus B'$. So, finally, splitting the node N_B into v_e plus the A -accessible vertices on one hand, and the B -accessible vertices on the other

hand, creates an edge e which separates the leaves of the tree into A and B .

In the second case, since there is an edge in G between every vertex in A' and every vertex in B' , then every marker vertex in the label of N is either A' -accessible or B' -accessible, but not both. Hence there is a split in N formed by A' -accessible and B' -accessible vertices. Hence N is a clique node. Assume a leaf $a \in A$ is in a connected component of $T - N$ containing vertices in B' . Then $a \notin A'$. Let v be the marker vertex of N associated with the edge of T adjacent to this connected component. Then v has to be A -accessible. Otherwise, the accessibility graph of the connected component would not be connected, contradicting Lemma 2.3. Since there are at least two connected components of $T - N$ containing vertices in B' , and since N is a clique, then there is an edge between a vertex in $A \setminus A'$ and a vertex in B' , which is a contradiction. So the connected components of $T - N$ contain either leaves in A or leaves in B but not both. And, finally, splitting the node N respectively with this partition creates an edge e which separates the leaves of the tree into A and B . \square

Proof of Theorem 2.9: With Proposition A.1, the list of splits of the graph determines the list of partitions of leaves of the tree induced by edge or node removals. It is an easy combinatorial property that this list determines completely the tree. Then the labels are necessarily determined by the graph structure. So, the whole graph-labelled tree is uniquely determined by the graph. \square