A Fully Dynamic Algorithm for the Recognition of P_4-sparse Graphs

Stavros D. Nikolopoulos[†] Leonidas Palios[†] Charis Papadopoulos[‡]

[†]*Department of Computer Science, University of Ioannina*
[‡]*Department of Mathematics, University of Ioannina*

P.O.Box 1186, GR-45110 Ioannina, Greece
`{stavros, palios, charis}@cs.uoi.gr`

Abstract: In this paper, we solve the dynamic recognition problem for the class of P_4-sparse graphs: the objective is to handle edge/vertex additions and deletions, to recognize if each such modification yields a P_4-sparse graph, and if yes, to update a representation of the graph. Our approach relies on maintaining the modular decomposition tree of the graph, which we use for solving the recognition problem. We establish properties for each modification to yield a P_4-sparse graph and obtain a fully dynamic recognition algorithm which handles edge modifications in $O(1)$ time and vertex modifications in $O(d)$ time for a vertex of degree d. Thus, our algorithm implies an optimal edges-only dynamic algorithm and a new optimal incremental algorithm for P_4-sparse graphs.

Keywords: fully dynamic algorithm, P_4-sparse graph, modular decomposition, recognition.

1 Introduction

A *dynamic graph* algorithm for a class Π of graphs is an algorithm that handles a series of on-line modifications (i.e., insertions or deletions of vertices or edges) on a graph in Π; if the modification results in a graph in Π, the algorithm performs it (updating an internal representation), otherwise it outputs `false` and does nothing. Such algorithms are categorized depending on the modification operations they support: an *incremental* (*decremental*) algorithm supports only vertex insertions (deletions); an *additions-only* (*deletions-only*) algorithm supports only edge additions (deletions); an *edges-only fully dynamic* algorithm supports both edge additions and edge deletions; a *fully dynamic* algorithm supports all edge as well as all vertex modifications.

Several authors have studied the dynamic recognition problem for graphs of specific families. Incremental recognitions algorithms have been proposed by Corneill *et al.* [3] for cographs and by Deng *et al.* [9] for connected proper interval graphs. Ibarra [15] has given an edges-only fully dynamic algorithm for chordal graph recognition which handles each edge operation in $O(n)$ time and an edges-only fully dynamic algorithm for split graph recognition which handles each edge operation in $O(1)$ time. Hell *et al.* [13] have given a fully dynamic algorithm for recognizing proper interval graphs which works in $O(d + \log n)$ time per modification, where d is the degree of a vertex in case of a vertex modification; Crespelle [5] has given a fully dynamic algorithm for recognizing interval graphs; based on the incremental algorithm for cographs [3], Shamir and Sharan [21] have developed a fully dynamic algorithm for the recognition of cographs, threshold graphs and trivially perfect graphs which handles edge modifications in $O(1)$ time and vertex modifications in $O(d)$ time; Crespelle and Paul have presented a fully dynamic algorithm for directed cographs which require $O(d)$ time if d arcs are involved

Graph Class	Edge Addition	Edge Removal	Vertex Insertion	Vertex Deletion
chordal	$O(n)$ [15]	$O(n)$ [15]	$O(dn)$ [1]	-
interval	$O(n)$ [5]	$O(n)$ [5]	$O(n)$ [5]	-
proper interval	$O(\log n)$ [13]	$O(\log n)$ [13]	$O(d + \log n)$ [13]	$O(d + \log n)$ [13]
split	$O(1)$ [15]	$O(1)$ [15]	$O(d)$ [12]	$O(d)$ [12]
permutation	$O(n)$ [7]	$O(n)$ [7]	$O(n)$ [7]	$O(n)$ [7]
directed cographs	$O(1)$ [6]	$O(1)$ [6]	$O(d)$ [6]	$O(d)$ [6]
cographs	$O(1)$ [21]	$O(1)$ [21]	$O(d)$ [3]	$O(d)$ [21]
trivially perfect	$O(1)$ [21]	$O(1)$ [21]	$O(d)$ [3]	$O(d)$ [21]
threshold	$O(1)$ [21]	$O(1)$ [21]	$O(d)$ [21]	$O(d)$ [21]
P_4-sparse	-	-	$O(d)$ [16]	-

Table 1: Summary of known results for fully dynamic recognition algorithms of several graph classes. For modifications involving a certain vertex v, we denote by d the degree of v.

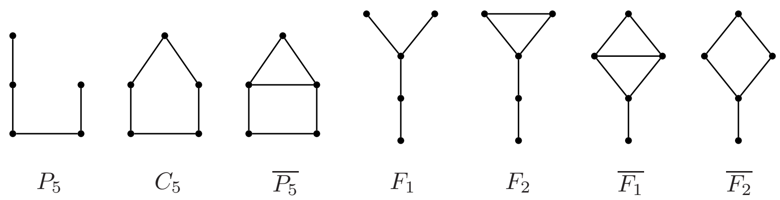


Figure 1: The seven forbidden subgraphs for the class of P_4-sparse graphs.

[6]; furthermore, the last two authors have developed a fully dynamic algorithm for permutation graphs which handles each modification in $O(n)$ time [7]. More recently two independent algorithms have been proposed for the dynamic recognition of distance-hereditary graphs [11, 23]. For the class of P_4-sparse graphs, an incremental algorithm for recognizing a P_4-sparse graph has been proposed by Jamison and Olariu [16] which handles the insertion of a vertex of degree d in $O(d)$ time. In Table 1 we summarize previously known results for the dynamic recognition of several graph classes.

Researchers have also considered the problem of the dynamic maintenance of the modular decomposition tree of a graph (the modular decomposition tree of a graph G is a unique (up to isomorphism) labeled tree which records all the partitions of the vertex set of G into modules and can be constructed in time and space linear in the size of the graph [4, 8, 18, 24]): Muller and Spinrad [19] have given an incremental algorithm for the modular decomposition, which handles each vertex insertion in $O(n)$ time; Corneil *et al.* [3] have given an optimal incremental algorithm for the recognition and modular decomposition of cographs, which handles the insertion of a vertex of degree d in $O(d)$ time.

Our work in this paper focuses on P_4-sparse graphs; the *P_4-sparse graphs* are defined as the graphs for which every set of five vertices induces at most one chordless path on four vertices [14] (Figure 1 depicts the 7 forbidden subgraphs for the class of P_4-sparse graphs). They are perfect and also perfectly orderable [14], and properly contain many graph classes, such as, the cographs, the P_4-reducible graphs, etc. (see [2, 16, 17]). The P_4-sparse graphs have received considerable attention in recent years and they find applications in applied mathematics and computer science (e.g., communications, transportation, clustering, scheduling, computational semantics) in problems that deal with graphs featuring "local density" properties. Indeed, the structure of P_4-sparse graphs incorporates such local density properties since they are graphs that are unlikely to have more than a few P_4s; we note that the notion of local density is often associated with the absence of P_4s.

In this paper, we describe a fully dynamic algorithm for the class of P_4-sparse graphs. Our algorithm maintains the modular decomposition tree of the graph; it checks whether the requested edge/vertex operations yield a P_4-sparse graph, and if yes, it updates the modular decomposition tree. Edge operations are handled in $O(1)$ time while vertex operations are handled in $O(d)$ time. As a result, we obtain an optimal edges-only dynamic algorithm and a new optimal incremental algorithm for P_4-sparse graphs.

As already mentioned, here we focus on maintaining a data structure of the given graph based on the modular decomposition tree. Such a representation is the ground of other dynamic recognition algorithms like for cographs [3, 21] and permutation graphs [7]. Thus it is expected that certain similarities occur between the dynamic algorithms that use the modular decomposition tree of the modified graph. Let us note that the modular decomposition tree of cographs (known as *cotree*) is quite restricted with the absence of one of the three types of label for the internal nodes of the tree. Since the class of P_4-sparse graphs properly contains cographs and it is unrelated with permutation graphs, the modular decomposition tree of a P_4-sparse graph generalizes that of a cograph whereas it differs to that of a permutation graph. Also we note that known vertex-incremental algorithm for P_4-sparse graphs [16] is not based on the modular decomposition tree and it seems non-trivial to extend the proposed algorithm in order to maintain the corresponding tree of the modified graph. Moreover it is known that the edge modifications on cographs have a local impact on the tree which enables an optimal running time [21]. We prove that a similar local impact occurs on P_4-sparse graphs, meaning that the two vertices incident to the modified edge cannot be far apart in the tree. For the vertex-incremental algorithm a marking process was used for the cograph recognition [3]. We are able to extend the marking process to the representation of a P_4-sparse graph so that it handles certain type of labelled node of the modular decomposition tree which is not present in the cotree of a cograph.

Our algorithm is based on a series of characterizations of the modified graph in order to provide necessary and sufficient conditions whenever the graph belongs to the class of P_4-sparse graphs. In some of the characterizations certain similarities occur on the stated conditions. The reason for giving all proofs in details is to provide a correct certificate of non-membership, i.e., a forbidden induced subgraph, when each of the corresponding condition does not hold. If G' is not P_4-sparse graph then it must contain one of the forbidden induced subgraphs depicted in Figure 1. However without the knowledge of the corresponding condition it is not clear to which certificate we are referring to and how this certain subgraph is obtained. Although our algorithm does not provide a certificate, it can be extended to do so, following our structural proofs. Therefore each of the characterization contributes in ensuring non-membership of the modified graph.

The paper is organized as follows. In Section 2 we establish the notation and related terminology, and we present background results on P_4-sparse graphs. In Section 3 we present our algorithmic techniques for handling edge modifications, while in Section 4 we present the case for handling vertex modifications. Final remarks and open problems are discussed in Section 5. A preliminary version of this work appeared in [20].

2 Theoretical Framework

Let G be a simple graph; we denote by $V(G)$ and $E(G)$, the vertex and edge set of G. The subgraph of G induced by a set $S \subseteq V(G)$ is denoted by $G[S]$. If a vertex u is adjacent to a vertex v, we say that u *sees* v, otherwise, we say that it *misses* v; more generally, a vertex set A sees (misses resp.) a vertex set B, if every vertex in A sees (misses resp.) every vertex in B. We denote by P_4 a chordless path on four vertices. The edge of a P_4 incident to the two vertices of degree two is called *middle edge* whereas the other two edges are called *wing edges*.

Let Π be a class of graphs. A *fully dynamic algorithm* for Π-recognition maintains a data structure of the current graph $G \in \Pi$ and supports the following operations.

- *Edge addition*: given two vertices $u, v \in V(G)$ which are non-adjacent in G, update the data structure if $G \cup \{uv\} \in \Pi$, or output *false* otherwise;
- *Edge removal*: given an edge $uv \in E(G)$, update the data structure if $G - \{uv\} \in \Pi$, or output *false* otherwise;

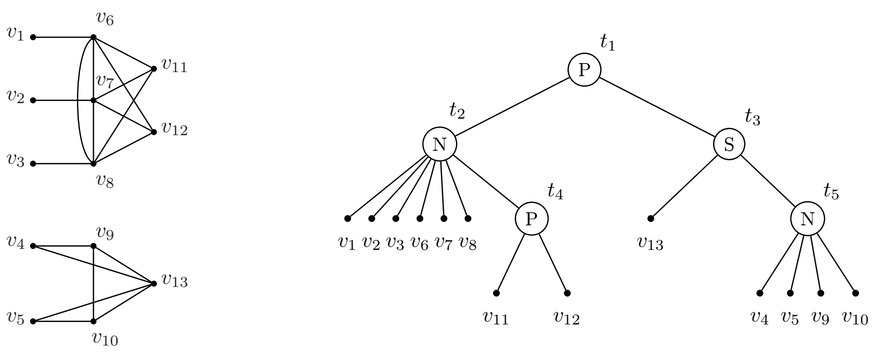


Figure 2: A disconnected P_4-sparse graph on 13 vertices and its md-tree.

- *Vertex insertion*: given a new vertex $v \notin V(G)$ adjacent to some vertices of G (possibly to none or all), update the data structure if $G \cup v \in \Pi$, or output *false* otherwise;
- *Vertex deletion*: given a vertex $v \in V(G)$, update the data structure if $G - v \in \Pi$, or output *false* otherwise.

After the execution of any such operation, the algorithm becomes ready to execute the next operation. Clearly (see also [21]), the addition (deletion) of a vertex does not reduce to the addition (deletion) of its incident edges; thus, vertex modifications must be handled separately from the edge modifications by the dynamic algorithm. However note that operations that handle vertex modifications can be used to support edge modifications as well in running time bounded by the time needed for a vertex removal and a vertex addition.

A graph class Π is called *c*omplement-invariant if $G \in \Pi$ implies $\overline{G} \in \Pi$. Further, we say that Π satisfies the *h*ereditary property if $G \in \Pi$ implies $G[S] \in \Pi$ for every $S \subseteq V$. We note that the class of P_4-sparse graphs is complement-invariant and satisfies the hereditary property.

Lemma 2.1. *Let G be a P_4-sparse graph. Then (i) G has the complement-invariant property, and (ii) for every $v \in G$, $G' = G - v$ is a P_4-sparse graph.*

2.1 Modular Decomposition and P_4-sparse Graphs

A subset M of vertices of a graph G is said to be a *module* of G, if every vertex outside M is either adjacent to all the vertices in M or to none of them. The empty set, the singletons, and the vertex set $V(G)$ are *trivial* modules and whenever G has only trivial modules it is called a *prime* (or *indecomposable*) *graph*. A module M of G is called a *strong module* if, for any module M' of G, either $M' \cap M = \emptyset$ or one module is included into the other. Furthermore, a module in G is also a module in $\overline{G}$.

The *modular decomposition* of a graph G is a linear-space representation of all the partitions of $V(G)$ where each partition class is a module. The *modular decomposition tree* $T(G)$ of the graph G (or *md-tree* for short) is a unique (up to isomorphism) labeled tree associated with the modular decomposition of G in which the leaves of $T(G)$ are the vertices of G and the set of leaves associated with the subtree rooted at an internal node induces a strong module of G (Figure 2). Thus, the md-tree $T(G)$ represents all the strong modules of G. It is known that for every graph G the md-tree $T(G)$ can be constructed in linear time [4, 8, 18, 24].

Let t be an internal node of the md-tree $T(G)$ of a graph G. We denote by $M(t)$ the module corresponding to t which consists of the set of vertices of G associated with the subtree of $T(G)$ rooted at node t. The node t is labeled by either P (for *parallel* module) if the subgraph $G[M(t)]$ is disconnected, S (for *series* module) if the complement of $G[M(t)]$ is disconnected, or N (for *neighborhood* module) otherwise. Let $u_1, u_2, \ldots, u_p$ be the children of the node t of $T(G)$. We denote by $G(t)$ the *representative graph* of the module $M(t)$ defined as follows: $V(G(t)) = \{u_1, u_2, \ldots, u_p\}$ and $u_i u_j \in E(G(t))$ if there exists edge $v_k v_\ell \in E(G)$ such that $v_k \in M(u_i)$ and $v_\ell \in M(u_j)$; by the definition of a module, if a vertex of $M(t_i)$ is adjacent to a vertex of $M(t_j)$ then every vertex of $M(t_i)$ is adjacent to every vertex of $M(t_j)$. Thus, $G(t)$ is isomorphic to the graph induced by a subset of $M(t)$ consisting of a single vertex from each maximal strong submodule of $M(t)$ in the modular decomposition of G. Depending on whether an internal node t of $T(G)$ is a P-, S-, or N-node, the following result holds (see also [10]):

- if t is a P-node, $G(t)$ is an edgeless graph;
- if t is an S-node, $G(t)$ is a complete graph;
- if t is an N-node, $G(t)$ is a prime graph.

In particular, for the class of P_4-sparse graphs, Giakoumakis and Vanherpe [10] showed that:

Lemma 2.2. *Let G be a graph and let $T(G)$ be its modular decomposition tree. The graph G is P_4-sparse iff for every N-node t of $T(G)$, $G(t)$ is a prime spider with a spider-partition (S, K, R) and no vertex of $S \cup K$ is an internal node in $T(G)$.*

A graph G is called a *spider* if the vertex set $V(G)$ of the graph G admits a partition into sets S, K, and R such that:

C1: $|S| = |K| \geq 2$, the set S is an independent (stable) set, and the set K is a clique;

C2: all the vertices in R are adjacent to all the vertices in K and to no vertex in S;

C3: there exists a bijection $f : S \longrightarrow K$ such that one of the following statements holds:

(i) for each vertex $v \in S$, $N(v) \cap K = \{f(v)\}$;
(ii) for each vertex $v \in S$, $N(v) \cap K = K - \{f(v)\}$.

The triple (S, K, R) is called the *spider-partition*. A graph G is a *prime spider* if G is a spider with $|R| \leq 1$. If the condition of case C3(i) holds, then the spider G is called a *thin spider*, whereas if the condition of case C3(ii) holds then G is a *thick spider*; note that the complement of a thin spider is a thick spider and vice versa. A prime spider with $|S| = |K| = 2$ is simultaneously thin and thick. Observe that in a spider graph every edge between vertices of S and K is a wing edge of a P_4 and every edge between vertices of K is a middle edge of a P_4.

2.2 Data Structure

As mentioned, our algorithm maintains the modular decomposition tree $T(G)$ of the P_4-sparse graph.

In order to facilitate our task, we store in each node of $T(G)$ additional information of constant size per node. More specifically, each node t of $T(G)$ stores

- its type (P, S, or N),
- a pointer to its parent, denoted by $p(t)$, and a pointer to its children,
- the number of its children, and
- auxiliary integer fields *counter* and *mark* initialized to 0.

Additionally, each N-node stores

- whether the corresponding spider is thin or thick;
- the independent set S and the clique K of the spider are stored in pairs of corresponding (through the function f) vertices,
- while there exists a separate pointer to R which is null if $R = \emptyset$ (there is no need to store the size $|S| = |K|$ as it is equal to $\lfloor c/2 \rfloor$, where c is the number of children of the N-node).

3 Edge Modifications

Here we show how to handle any edge addition and edge removal of a P_4-sparse graph.

3.1 Adding an Edge

Let uv be the edge to be added and let $G' = G \cup \{uv\}$. For the two vertices u, $v \in G$ we denote by t_{uv} the least common ancestor of u and v in $T(G)$. Since u, v are non-adjacent in G, node t_{uv} is either a P-node or an N-node.

Let us first discuss the modifications needed in order to update properly $T(G)$. We show that we need to update either the subtree rooted at t_{uv} or the subtree rooted at $p(t_{uv})$. In G' every vertex of $V(G') \setminus M(t_{uv})$ either sees $M(t_{uv})$ or misses $M(t_{uv})$ and, thus, $M(t_{uv})$ remains a module of G'. Let T'_{uv} be the modular decomposition tree of $G'[M(t_{uv})]$. Then $T(G')$ is obtained from $T(G)$ by substituting the subtree rooted at t_{uv} by T'_{uv}. However if $p(t_{uv})$ has the same P or S label with the root of T'_{uv} we need to be more careful. Because we add an edge in $G[M(t_{uv})]$, if the root of T'_{uv} is a P-node then t_{uv} is also a P-node and $p(t_{uv})$ is not a P-node. Thus if both $p(t_{uv})$ and the root of T'_{uv} have the same label then it implies that they are both S-nodes. The latter occurs when u or v sees every vertex of $M(t_{uv})$ in G'. Assume without loss of generality that u sees every vertex of $M(t_{uv})$ in G'. Since $G[M(t_{uv})]$ is disconnected (rooted at a P-node) and $G'[M(t_{uv})]$ is connected (rooted at an S-node), $G'[M(t_{uv})] - u$ is disconnected or $G'[M(t_{uv})] - u$ has only one vertex v. Thus the the modular decomposition tree $T'_{uv} - u$ of $G'[M(t_{uv})] - u$ is rooted at a P-node or contains only a leaf-vertex. Then we substitute the subtree rooted at t_{uv} by $T'_{uv} - u$ and place u as a child of $p(t_{uv})$. Therefore the addition of the edge uv in G results in updating one of the subtrees rooted at t_{uv} or $p(t_{uv})$.

Let t_u and t_v be the children of t_{uv} such that $M(t_u)$ and $M(t_v)$ contain the vertices u and v, respectively. Note that if $|M(t_u)| = 1$ (resp. $|M(t_v)| = 1$) then $t_u = u$ (resp. $t_v = v$). Without loss of generality, we make the following assumption:

Assumption 3.1. *We assume that* $|M(t_v)| \geq |M(t_u)|$.

We distinguish three cases, namely, (i) $|M(t_u)| \geq 2$, (ii) $|M(t_u)| = 1$ and t_{uv} is a P-node, and (iii) $|M(t_u)| = 1$ and t_{uv} is an N-node; we prove the following lemmas.

Lemma 3.2. *Let* $|M(t_u)| \geq 2$. *Then* G' *is a* P_4*-sparse graph if and only if* t_{uv} *is a P-node and* $|M(t_u)| = |M(t_v)| = 2$.

Proof. Since $|M(t_v)| \geq |M(t_u)| \geq 2$, the node t_{uv} cannot be an N-node because t_u and t_v are internal nodes and at most one child of any N-node is an internal node (not a leaf) in $T(G)$ by Lemma 2.2. Thus, t_{uv} is a P-node; then it follows that the subgraphs $G[M(t_u)]$ and $G[M(t_v)]$ are both connected.

For the "if"-part of the lemma let $M(t_u) = \{u, u'\}$ and $M(t_v) = \{v, v'\}$. Observe that in G' we create a new P_4 by adding its middle edge uv. We describe the modifications applied in $T(G')$. We create a new subtree rooted at an N-node γ having children u, u', v, v'. The spider partition (S, K, R) of $G(\gamma)$ corresponds to $S = \{u', v'\}$, $K = \{u, v\}$, and $R = \emptyset$. If t_{uv} has exactly two children in $T(G)$ then t_{uv} is replaced by γ in $T(G')$; otherwise, t_u and t_v are removed from children of t_{uv} and γ becomes a child of t_{uv} in $T(G')$. Hence the resulting graph G' is indeed P_4-sparse. For the "only if"-part, we have that G' is P_4-sparse and assume for contradiction that at least one of $M(t_u), M(t_v)$ has 3 elements; then, Assumption 3.1 implies that $|M(t_v)| \geq 3$. The connectivity of $G[M(t_u)]$ and $G[M(t_v)]$ implies that there exist vertices $u' \in M(t_u)$ and $v' \in M(t_v)$ such that $uu', vv' \in E(G)$. Then, by adding the edge uv in G, the resulting graph G' contains the P_4 $u'uvv'$. Since $G[M(t_v)]$ is connected and $|M(t_v)| \geq 3$, there exists a vertex x in $M(t_v)$ such that x sees at least one of v, v'. But then, the five vertices u', u, v, v', x induce in G' one of the following graphs: P_5, F_1, or F_2; thus, G' is not P_4-sparse, a contradiction. ∎

For the following observe that if statement (i) holds then we do not create a new P_4 in G'; this corresponds to the cograph recognition case as stated in [21]. If statement (ii) holds then we create a new P_4 in G' by adding a wing edge.

Lemma 3.3. *Let $M(t_u) = \{u\}$, t_{uv} be a P-node, and suppose that the path from t_v to $p(v)$ in the md-tree $T(G)$ does not contain any N-node. If G' is a P_4-sparse graph then the following two properties are satisfied:*

(i) there exists at least one vertex in $M(t_v)$ which sees all the other vertices in $M(t_v)$;

(ii) if vertex v misses at least one vertex in $M(t_v)$, then there exists exactly one vertex, say, x, in $M(t_v)$ which sees all the other vertices in $M(t_v)$, v misses exactly one vertex, say, y, in $M(t_v)$, and y only sees x.

Proof. (i) Suppose for contradiction that there is no vertex in $M(t_v)$ that sees all the other vertices in $M(t_v)$. Since t_{uv} is a P-node and there are no N-nodes from t_v to $p(v)$, t_v is an S-node. Furthermore, because there is no vertex $x \in M(t_v)$ that sees $M(t_v) - \{x\}$, t_v has no children that are leaves of the md-tree $T(G)$. Thus, t_v has at least two children that are internal nodes of $T(G)$; let t be the child of t_v that is an ancestor of v and let t' be another child of t_v. Both t and t' are not S-nodes, and there exist $p \in M(t) - \{v\}$ and $q, r \in M(t')$, such that v, p are not adjacent and q, r are not adjacent in G. Then, the vertices u, v, p, q, r induce an $\overline{F}_2$ in G', a contradiction. Therefore, there exists at least one vertex in $M(t_v)$ that sees all the other vertices in $M(t_v)$.

(ii) We first show that there is no vertex $x' \in M(t_v) - \{x\}$ that sees all other vertices in $M(t_v)$. If there were such a vertex x', then, the five vertices u, v, x, y, and x' induce in G' the graph $\overline{F}_1$, a contradiction. Thus, only vertex x in $M(t_v)$ sees all other vertices in $M(t_v)$.

Next, suppose that there exists another vertex $y' \in M(t_v) - \{y\}$ such that v misses y'. Then, the five vertices u, v, x, y, and y' induce in G' the graphs F_1 or F_2, a contradiction. Therefore, v misses exactly one vertex in $M(t_v)$, the vertex y. Moreover, if y saw a vertex, say, $z \in M(t_v)$, other than x, then the five vertices u, v, x, y, and z would induce in G' the graph $\overline{F}_1$, a contradiction again. ∎

Suppose now that the path from t_v to $p(v)$ contains at least one N-node t of $T(G)$. Recall that the representative graph $G(t)$ is a prime spider and let (S, K, R) be its spider partition. Note that $v \in M(t)$ and thus v belongs to the set S, or to the set K, or if $R = \{r\}$ to the set $M(r)$.

Lemma 3.4. *Let $M(t_u) = \{u\}$, t_{uv} be a P-node, and suppose that the path from t_v to $p(v)$ in the md-tree $T(G)$ contains at least one N-node t. Let (S, K, R) be the spider partition of $G(t)$ with $R = \{r\}$. If G' is a P_4-sparse graph then the following three properties are satisfied:*

(i) $v \in M(r)$ and v sees all other vertices in $M(r)$;

(ii) the path from t_v to $p(v)$ contains exactly one N-node t;

(iii) $t = t_v$ and $G(t_v)$ is a thin spider.

Proof. (i) Suppose for contradiction that the vertex v belongs to the set S of the prime spider $G(t)$. Since $|S| = |K| \geq 2$, there exists a vertex $v' \in S - \{v\}$. If the P_4 of $G(t)$ to which v, v' belong is $vyy'v'$, then the addition of the edge uv implies that G' contains the P_5 $uvyy'v'$, a contradiction. Now consider that $v \in K$; let $v' \in K - \{v\}$ and let $zvv'z'$ be the P_4 of $G(t)$ to which v, v' belong. Then, the five vertices z, v, v', z', u induce in G' the graph F_1, a contradiction again. Thus, since the graph G' is P_4-sparse, it must hold that $v \in M(r)$.

Suppose now that $v \in M(r)$ and let $z \in M(r)$ be such that v misses z. If $x \in S$ and $y \in K$ such that x, y are adjacent in G, then by adding the edge uv, the five vertices u, v, x, y, and z induce the graph F_1 in G'.

(ii) Suppose that there existed another N-node, say, t', in the path from t_v to $p(v)$ and let (S', K', R') be the spider partition of $G(t')$ with $R' = \{r'\}$. Suppose without loss of generality that t' is higher in the tree $T(G)$ than t. Then, according to statement (i) above, v would belong to $M(r) \subset M(r')$; yet, v would miss all the vertices of $S \subset M(r')$, in contradiction to statement (i).

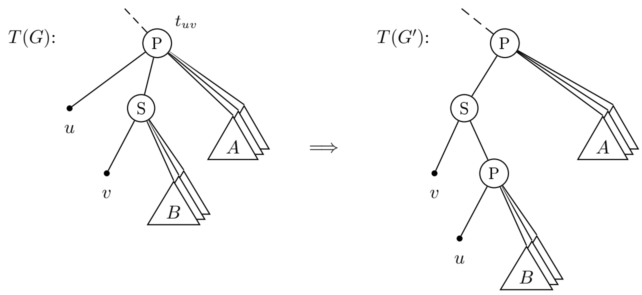


Figure 3: Illustrating case (i) of Corollary 3.5 and the corresponding updates of the md-tree. If there is no subtree labelled A in $T(G)$ then the top-most P-node in the subtree of $T(G')$ is not needed and the subtree is now rooted at the S-node which is the parent of v.

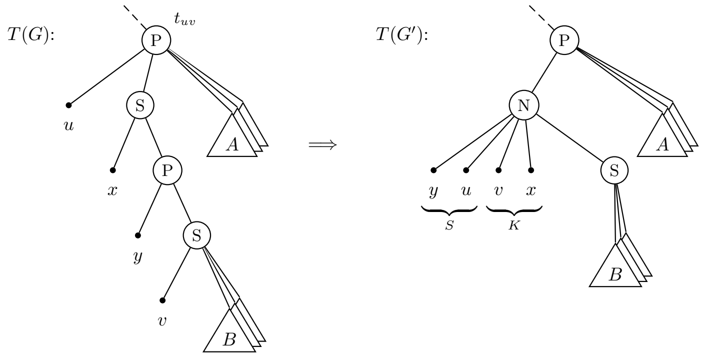


Figure 4: Illustrating case (ii) of Corollary 3.5 and the corresponding updates of the md-tree. If there is no subtree labelled A in $T(G)$ then the P-node in the subtree of $T(G')$ is not needed and the subtree is now rooted at the N-node which is the parent of v.

(iii) Suppose now that $t \neq t_v$. Since t_{uv} is a P-node and $t \neq t_v$, it follows that t_v is an S-node (if t_v were an N-node, then the path would contain two N-nodes). Then, at least one vertex $z \in M(t_v)$ sees all the vertices of $M(t)$. Let $x, y \in S$ be two vertices of the spider $G(t)$. By adding the edge uv in G, the five vertices u, v, x, y, and z of G' would induce the graph F_1 in G', a contradiction. Moreover, if $G(t_v)$ were a thick spider, then there would exist $x_1, x_2 \in S$ and $y \in K$ such that y would see both x_1 and x_2. By statement (i) above, $v \in M(r)$. Then, the addition of the edge uv would imply that the vertices x_1, x_2, y, v, u would induce the graph F_1 in G'. ∎

Then, from Lemmas 3.3 and 3.4, we have:

Corollary 3.5. *Let $|M(t_u)| = 1$ (i.e., $M(t_u) = \{u\}$) and suppose that t_{uv} is a P-node. Then G' is a P_4-sparse graph if and only if one of the following (mutually exclusive) cases holds:*

(i) vertex v sees all the vertices in $M(t_v) - \{v\}$;

(ii) vertex v misses exactly one vertex $y \in M(t_v)$ such that y sees only one vertex $x \in M(t_v)$, and only the vertex x sees all the other vertices in $M(t_v)$;

(iii) vertex v misses $\ell > 1$ vertices in $M(t_v)$ such that $G(t_v)$ is a thin spider (S, K, R) with $|S| = |K| = \ell$, $R = \{r\}$ and the vertex v belongs to the set $M(r)$ and sees all the other vertices in $M(r)$.

Proof. For the "if"-part let us construct an md-tree for G' having the properties described in Lemma 2.2. If Case (i) holds then v is a child of the S-node t_v in $T(G)$. In the md-tree of G', a new P-node is

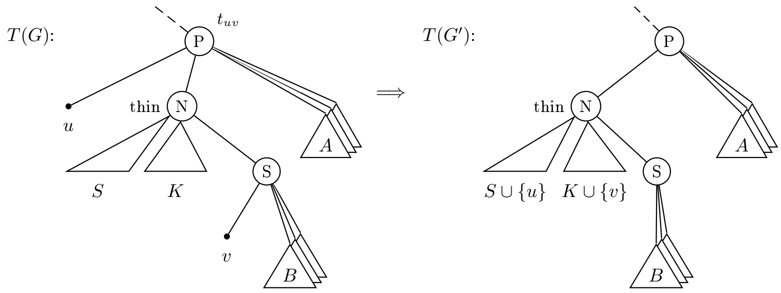


Figure 5: Illustrating case (iii) of Corollary 3.5 and the corresponding updates of the md-tree. If there is no subtree labelled A in $T(G)$ then the P-node in the subtree of $T(G')$ is not needed and the subtree is now rooted at the N-node which is the parent of v.

inserted as a child of t_v having children u and an S-node γ. The previous children besides v of t_v now point to node γ keeping their adjacencies they had in G (see Figure 3; note that this case corresponds to the cograph characterization as stated in [21]). The rest of the cases follow in a similar manner as shown in Figures 4–5.

For the "only if"-part, we have that G' is P_4-sparse. Then, if the path from t_v to $p(v)$ in the md-tree $T(G)$ contains no N-nodes, then by Lemma 3.3 (statement (i)), there exists a vertex in $M(t_v)$ which sees all other vertices in $M(t_v)$. If v is such a vertex, then we get Case (i). If v is not so, i.e., v misses at least one vertex in $M(t_v) - \{v\}$, then Lemma 3.3 (statement (ii)), implies that Case (ii) holds. If now the path from t_v to $p(v)$ in the md-tree $T(G)$ contains N-nodes, then by Lemma 3.4, we have that t_v is the unique N-node in the path and $G(t_v)$ is a thin spider. Furthermore, if (S, K, R) is the spider partition of $G(t)$, then $R = \{r\}$ and $v \in M(r)$ and sees all other vertices in $M(r)$. Thus, v would miss only the vertices in S, where $|S| > 1$; this is Case (iii). ∎

Lemma 3.6. *Let $|M(t_u)| = 1$ (i.e., $M(t_u) = \{u\}$) and suppose that t_{uv} is an N-node with (S, K, R) being the spider partition of $G(t_{uv})$. Then G' is a P_4-sparse graph if and only if either $S = \{u, v\}$ and $R = \emptyset$ or $u \in S$, $v \in K$, and $G(t_{uv})$ is a thick spider.*

Proof. Let us first show that if one of the two conditions of the statement holds then G' is a P_4-sparse graph. In order to show that G' is a P_4-sparse graph we construct an md-tree $T(G')$ for G' with the properties described in Lemma 2.2. If $S = \{u, v\}$ and $R = \emptyset$ then the four vertices of $G(t_{uv})$ create a chordless cycle and the subtree of t_{uv} is replaced with the subtree shown in the top part of Figure 6. Let γ be the new S-node having the two children that correspond to the subtrees of u and v. If the parent of t_{uv} exists and is an S-node then the two children of γ become children of $p(t_{uv})$ and γ is removed since two adjacent S-nodes cannot exist in $T(G')$. Otherwise, if t_{uv} is the root of $T(G)$ or $p(t_{uv})$ is an P- or N-node then γ becomes a child of or $p(t_{uv})$ in $T(G')$. For the case when $u \in S$, $v \in K$, and $G(t_{uv})$ is a thick spider we work in a similar manner as shown in the two other parts of Figure 6 in which we distinguish the modified $T(G')$ according to whether $|S| = 2$ or $|S| > 2$.

Next assume that G' is a P_4-sparse graph. Then G' does not contain one of the graphs in Figure 1 as induced subgraphs. The definition of the spider implies that the cases to consider are for u, v to belong both to S, or to S and K, or if $R = \{r\}$ to S and $M(r)$.

- $u, v \in S$: Let $u', v' \in K$ such that $uu'v'v$ is a P_4 of G; then, G' contains the chordless cycle $uu'v'v$. If $R = \{r\}$ then the vertices u, v, u', v' and any vertex in $M(r)$ induce a $\overline{P}_5$ in G'; thus, $R = \emptyset$. If $|S| = |K| > 2$, then if the spider is thin, the vertices u, v, u', v', y, where $y \in K - \{u', v'\}$ induce a $\overline{P}_5$, whereas if the spider is thick, the vertices u, v, u', v', z, where $z \in S - \{u, v\}$ induce a $\overline{P}_5$ in G'.

- $u \in S$, $v \in K$: Suppose that $G(t_{uv})$ is a thin spider such that $|S| > 2$ (note that the spiders with $|S| = 2$ are also considered thick); R may or may not be $\emptyset$. Then, $v \neq f(u)$. Let $z \in K$ be

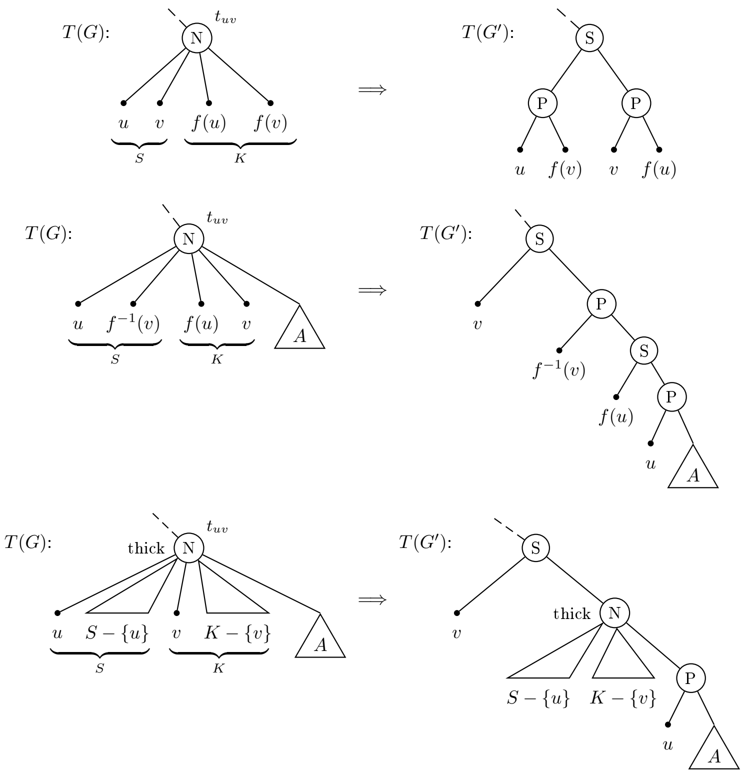


Figure 6: Illustrating the cases of Lemma 3.6 and the corresponding updates of the md-tree.

such that $z \neq v$ and $z \neq f(u)$; then, the vertices $u, v, f(u), z, f^{-1}(z)$ induce a graph $\overline{F}_1$ in G'.

- $u \in S$ and $v \in M(r)$: Let $x \in S - \{u\}$. If $z, z' \in K$ are the vertices such that $uzz'x$ is a P_4 in $G(t_{uv})$ (and in G), then the vertices u, v, x, z, z' induce an $\overline{F_1}$ in G', which thus is not P_4-sparse.

Therefore if G' is a P_4-sparse graph then the conditions of the statement hold and we conclude the proof. ∎

Due to the previous results, one can notice that the addition of an edge in a P_4-sparse graph can never create a new thick spider with $|S| \geq 3$. We conclude this section with the following result.

Theorem 3.7. *The insertion of an edge in a P_4-sparse graph can be handled in $O(1)$ time.*

Proof. We apply Lemma 3.2, Corollary 3.5, and Lemma 3.6, that correspond to different cases depending on the cardinality of $M(t_u)$ (whether it is a singleton set) and the label of t_{uv} (P-node or an N-node). In fact we show that checking the corresponding cases and properly updating $T(G)$ can be done through traversing $T(G)$ in constant time.

More precisely Lemma 3.2 reads in terms of $T(G)$ as follows: u and v have the same grand-parent which is a P-node and both parents of u and v have exactly two children that are leafs in $T(G)$. Thus checking the corresponding statement requires going up from u, v by at most 2 levels which can be done in $O(1)$ time. Further obtaining $T(G')$ can be done through updating the corresponding fields of constant number of nodes.

Next, we express the cases of Corollary 3.5 in terms of $T(G)$ as follows:

(i) $p(v)$ is not an N-node and either $p(v)$ or the grand-parent of v coincides with $p(u)$;

(ii) $p(v)$ is not an N-node and either the great grand-parent of v coincides with $p(u)$, and the grand-parent of v has exactly two children one of which is leaf, or the great great grand-parent of v coincides with $p(u)$, and both the grand-parent and the great grand-parent of v have exactly two children one of which is leaf.

(iii) either $p(v)$ is an N-node such that $G(p(v))$ is a thin spider (S, K, R) with $R = \{v\}$ and the grand-parent of v coincides with $p(u)$, or $p(v)$ is an S-node, the grand-parent of v is an N-node corresponding to a thin spider, and the great grand-parent of v coincides with $p(u)$.

Each case requires going up from v by at most 2 levels (case (i)) or 4 levels (case (ii)) or 3 levels (case (iii)). All the necessary conditions are checked through the additional information stored in each node of $T(G)$. We need to be careful when we update properly $T(G)$ because of the movement of v that may result in a node having exactly one child in $T(G')$ (this corresponds to the root of the subtree labelled B at Figures 3,4,5). However this can be done in constant time as it corresponds in updating the parent-pointers of constant number of nodes. Thus it is not difficult to see that updating $T(G)$ can be done in constant time.

Regarding Lemma 3.6, the corresponding cases translate in terms of $T(G)$ as follows: u and v have the same parent which is an N-node with spider partition (S, K, R) such that either $S = \{u, v\}$ and $R = \emptyset$, or $u \in S$, $v \in K$, and $G(p(u))$ is a thick spider. Thus it requires on checking the parent nodes of u, v and the additional information stored at the N-node. Updating $T(G)$ can be done by introducing properly constant number of nodes with the corresponding information as shown in Figure 6.

Hence checking whether any of the corresponding cases holds requires going up from v, u by at most 4 levels from v and possibly from u which can be done in $O(1)$ time using the parent-pointers and by spending $O(1)$ time per level for verification purposes. Additionally, updating the md-tree can also be done in $O(1)$ time. Therefore, the theorem holds. ∎

3.2 Removing an Edge

In order to handle edge removal, we take advantage of the complement-invariant property of the class of P_4-sparse graphs (see Lemma 2.1). In fact we will apply the algorithm for the edge addition through Theorem 3.7 on the complement of G: by the complement-invariant property, an edge uv can be added in $\overline{G}$ if and only if the edge uv can be removed from G. In order to avoid storing $\overline{G}$ we will use $T(G)$ as a different interpretation for $T(\overline{G})$. Indeed $T(\overline{G})$ is obtained from $T(G)$ by exchanging the roles of P- and S-nodes, and for each N-node with spider partition (S, K, R) we exchange the roles of thin and thick spiders, and swap the sets S and K. Thus by using a different interpretation of $T(G)$ in Theorem 3.7, edge deletions for P_4-sparse graphs can also be handled in $O(1)$ time.

Therefore we obtain the following result.

Theorem 3.8. *There is an optimal edges-only fully dynamic algorithm for recognizing P_4-sparse graphs and maintaining their modular decomposition tree, which handles each edge modification in $O(1)$ time.*

4 Vertex Modifications

First we handle vertex insertions and, then, we handle vertex deletions.

4.1 Adding a Vertex

Let G be a P_4-sparse graph and a vertex $x \notin V(G)$ which is adjacent to d vertices in $V(G)$, where $d \in \{0, 1, \ldots, |V(G)|\}$. In this section, we show how to recognize if the graph G' with vertex set $V(G) \cup \{x\}$ is a P_4-sparse graph, and if so, we show how to obtain the md-tree $T(G')$ of G' from the md-tree $T(G)$ in $O(d)$ time. Let us classify the internal nodes of the md-tree $T(G)$ with respect to the vertex x into the following three categories: an internal node t is *x-fully-adjacent*, *x-partly-adjacent*, *x-non-adjacent* iff x is adjacent to all, some but not all, and none, respectively, of the vertices in the module $M(t)$. The above classification is extended to leaf-nodes: a leaf-node a is *x-fully-adjacent* or *x-non-adjacent* iff x is adjacent or non-adjacent respectively to a. For the x-fully-adjacent nodes of $T(G)$, we have the following observation:

Observation 4.1. *The x-fully-adjacent nodes form a forest of at most d subtrees of $T(G)$ whose total number of nodes (i.e., internal and leaf) is less than $2d$, where d is the number of vertices of G which are adjacent to x.*

Proof. The observation follows from the fact that the forest of x-fully-adjacent nodes has d leaves and that every internal node in $T(G)$ and in this forest has at least two children. ∎

In turn, for the x-partly-adjacent nodes, we show the following properties:

P1: if an internal node t of the md-tree $T(G)$ is x-partly-adjacent, then all its ancestors in $T(G)$ are x-partly-adjacent;

P2: for every x-partly-adjacent P-node t_P of $T(G)$, the subgraph of G induced by the module $M(t_P)$ contains two non-adjacent vertices a, b such that a is adjacent and b is not adjacent to x;

P3: for every x-partly-adjacent S-node t_S of $T(G)$, the subgraph of G induced by the module $M(t_S)$ contains an edge ab such that a is adjacent and b is not adjacent to x;

P4: for every x-partly-adjacent N-node t_N of $T(G)$, the subgraph of G induced by the module $M(t_N)$ contains both an edge ab such that a is adjacent and b is not adjacent to x and a pair of non-adjacent vertices a', b' such that a' is adjacent and b' is not adjacent to x.

Property P1 holds by the definition of x-partly-adjacent nodes. Let us show that Property P2 holds. The vertices of $M(t_P)$ can be partitioned into two non-empty sets according to whether they are adjacent or not to x. Since the complement of $G[M(t_P)]$ is connected (t_P is a P-node), there must be a non-edge in $G[M(t_P)]$ between the two sets of the partition. The endpoints of the particular non-edge correspond to the vertices a and b. Properties P3–P4 follow by similar arguments, taking into account that the graph induced by the module of an S-node is a connected graph, and the graph induced by the module of an N-node is connected and its complement is also connected.

Additionally, the following very important property holds:

Theorem 4.2. *For any two x-partly-adjacent nodes of $T(G)$, the graph G' is P_4-sparse only if one of them is an ancestor of the other.*

Proof. Suppose that $T(G)$ contains two x-partly-adjacent nodes t, t' such that none is an ancestor of the other. Then, t, t' are internal nodes of $T(G)$ and let t_i be the least common ancestor of t, t', and t_j and t_k be the children of t_i which are ancestors of t and t' respectively. Clearly, by Property P1, t_j and t_k are x-partly-adjacent nodes. Additionally, the node t_i is either a P-node or an S-node (recall that at most one child of an N-node is an internal node). Thus, we distinguish the following two cases:

- *the node t_i is a P-node*: Then, t_j, t_k are either S- or N-nodes; in either case, there are vertices $a_j, b_j \in M(t_j)$ and $a_k, b_k \in M(t_k)$ such that in G, a_j, b_j are adjacent, a_k, b_k are also adjacent, and x is adjacent to a_j, a_k but not to b_j, b_k (see Properties P3, P4). But then, G' would contain the P_5 $b_j a_j x a_k b_k$, and thus would not be P_4-sparse.

- *the node t_i is an S-node*: This case is the complement version of the previous case. The nodes t_j, t_k are either P- or N-nodes; in either case, there are vertices $a_j, b_j \in M(t_j)$ and $a_k, b_k \in M(t_k)$ such that in G, a_j, b_j are non-adjacent, a_k, b_k are also non-adjacent, and x sees a_j, a_k and misses b_j, b_k (see Properties P2, P4). But then, G' would not be P_4-sparse as it would contain the $\overline{P}_5$ induced by a_j, b_j, x, a_k, b_k.

∎

Theorem 4.2 implies that for G' to be P_4-sparse all the x-partly-adjacent nodes form a path ρ_x with at most one node per level of $T(G)$. Since the root of $T(G)$ is an x-partly-adjacent node if there exist any such nodes, let $\rho_x = t_0 t_1 \cdots t_k$ where t_0 is the root of $T(G)$ and t_k is the x-partly-adjacent node farthest away from the root. Additionally, Theorem 4.2 implies that for each node t_i, $0 \leq i < k$, each of t_i's children, other than t_{i+1}, is either x-fully-adjacent or x-non-adjacent; for the node t_k, each of t_k's children is either x-fully-adjacent or x-non-adjacent and there is at least one child of each kind.

For the x-partly-adjacent N-nodes, we can also show the following:

Lemma 4.3. *Let t be an x-partly-adjacent N-node of $T(G)$ whose corresponding spider partition of $M(t)$ is (S, K, R), and suppose that the vertex x is adjacent to a vertex in $S \cup K$. Then, the graph G' is P_4-sparse only if x sees $S \cup K$, or sees K and misses S.*

Proof. First, suppose that x sees $k \in K$. Then we show that x sees every $k' \in K$. Let $skk's'$ be the (unique) P_4 of the spider which has kk' as an edge. If x misses k' then the vertices x, k, k', s, s' induce an F_1 (if x misses both s, s'), or an F_2 (if x sees s but misses s'), or an $\overline{F}_2$ (if x sees s' but misses s), or a $\overline{P}_5$ (if x sees both s, s'). Thus x either sees K or misses it.

In a similar fashion we show that either x sees S or misses it. Suppose that x sees $s \in S$ and does not see $s' \in S$. Then, if $skk's'$ is the (unique) P_4 of the spider containing both s and s', the vertices x, k, k', s, s' induce an $\overline{F}_1$ if x is adjacent to both k, k', or a P_5 if x is adjacent neither to k nor to k' (note that in light of our result for K, we do not need to consider the case where x is adjacent to exactly one of k, k'). Thus, x either sees S or misses it.

Finally we show that if x misses K and sees S then G' is not P_4-sparse. Let $skk's'$ be a P_4 of the spider, where $k, k' \in K$ and $s, s' \in S$. If x missed k, k' but saw s, s' then the vertices x, k, k', s, s' would induce a C_5 in G', a contradiction. ∎

Let us consider the partition of the vertex set $M(t_0) - M(t_k) \subset V(G)$ into the following four sets:

$$V_P = \bigcup_{\substack{t_i \text{ is a P-node} \\ 0 \leq i < k}} \left(M(t_i) - M(t_{i+1})\right), \qquad V_S = \bigcup_{\substack{t_i \text{ is an S-node} \\ 0 \leq i < k}} \left(M(t_i) - M(t_{i+1})\right),$$

$$V_{N_S} = \bigcup_{\substack{t_i \text{ is an N-node} \\ 0 \leq i < k}} S(t_i), \qquad V_{N_K} = \bigcup_{\substack{t_i \text{ is an N-node} \\ 0 \leq i < k}} K(t_i),$$

where for an N-node t_i, $S(t_i)$ and $K(t_i)$ are the independent set and the clique of the spider induced by the module $M(t_i)$. Then, every vertex in V_P (in V_S resp.) is non-adjacent (adjacent resp.) to the vertices in $M(t_k)$ since their least common ancestor t_i in $T(G)$ is a P-node (S-node resp.), while the structural properties of a spider imply that every vertex in $K(t_j)$ ($S(t_j)$ resp.) for an N-node t_j is adjacent (non-adjacent resp.) to the vertices in $M(t_k)$.

Our vertex-addition procedure relies on the following lemmas:

Lemma 4.4. *Suppose that the x-partly-adjacent nodes of the md-tree $T(G)$ lie on a path $t_0 t_1 \cdots t_k$, where t_0 is the root of $T(G)$ and t_i is the parent of t_{i+1} for each $i = 0, 1, \ldots, k-1$. If t_k is a P-node then G' is P_4-sparse if and only if one of the following four (mutually exclusive) cases holds:*

(i) Vertex x sees V_S and V_{N_K}, and misses V_P and V_{N_S}.

(ii) Vertex x sees V_S, V_{N_K}, and exactly one vertex, say, y, in V_P, and misses V_{N_S} where

(ii.1) vertex y is a child of node t_{k-2} (which is a P-node),

(ii.2) node t_{k-1} is an S-node with two children, the node t_k and one vertex, say, u (which is adjacent to x), and

(ii.3) *vertex x sees all the vertices in $M(t_k)$ except for a single vertex, say, b, which is a child of t_k.*

(iii) *Vertex x sees V_{N_K}, all but one vertex, say, z, in V_S, and misses V_P and V_{N_S} where*

(iii.1) *vertex z is a child of node t_{k-1} (which is an S-node), and*

(iii.2) *node t_k has two children a, b, which are leaf-nodes such that a is adjacent and b is non-adjacent to x.*

(iv) *The node t_{k-1} is an N-node corresponding to a thick spider with independent set $S(t_{k-1})$, vertex x sees V_S, V_{N_K}, $S(t_{k-1})$, and all but one vertex, say, b (which is a child of t_k), in $M(t_k)$, and misses V_P and $V_{N_S} - S(t_{k-1})$.*

Proof. It is not difficult to see that the graph G' is P_4-sparse if Case (i) of the lemma holds: in the md-tree of G', x and the x-fully-adjacent children of t_k in $T(G)$ are children of an S-node which is a child of t_k (see Figure 7 (a)). Similarly, for the remaining cases, Figure 7 gives the md-tree $T(G')$ (it is easy to check the adjacencies), which establishes that the graph G' is P_4-sparse in these cases as well. Thus, we need to show that if G' is P_4-sparse exactly one of Cases (i)–(iv) holds.

Since the node t_k is an x-partly-adjacent P-node then by Property P2 there exist two vertices $a, b \in M(t_k)$ which are non-adjacent in G and such that x is adjacent to a and non-adjacent to b. First, we note that, for G' to be P_4-sparse:

A1: *x must be adjacent to the entire V_{N_K} and to all but at most one vertex in V_S*: if x were not adjacent to vertices $y, y' \in V_S \cup V_{N_K}$, then the vertices x, a, b, y, y' would induce in G' either an $\overline{F}_1$ or an $\overline{F}_2$ (see Figure 1) depending on whether y, y' are adjacent or not, and thus G' would not be P_4-sparse. Thus, x is non-adjacent to at most one vertex in $V_S \cup V_{N_K}$. Since the clique of a spider is of size at least 2, x is adjacent to at least one vertex of the clique. Then by Lemma 4.3 x sees all of V_{N_K}. Therefore if x does not see a vertex $y \in V_S \cup V_{N_K}$, then $y \in V_S$.

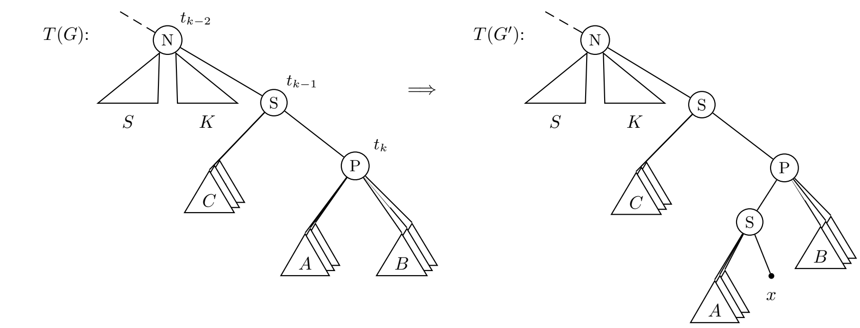
A2: *x must be adjacent to at most one vertex in V_P*: suppose that x were adjacent to vertices $z, z' \in V_P$; since the parent of t_k is either an S- or an N-node, there exists a vertex y which is adjacent to a, b and is non-adjacent to z, z'. Then, if y is non-adjacent to x, the vertices z, x, a, y, b would induce a P_5. If y is adjacent to x, the vertices b, y, x, z, z' would induce an F_1 ($zz' \notin E(G)$), or an F_2 ($zz' \in E(G)$).

A3: *x must miss the independent sets of all the N-nodes in the subpath $t_0 t_1 \cdots t_{k-2}$*: suppose that x were adjacent to a vertex z belonging to the independent set $S(t_i)$ of the spider associated with t_i ($0 \le i \le k-2$); then, there exists $k \in K(t_i)$ such that k is non-adjacent to z, and since x sees V_{N_K}, k is adjacent to x as well; moreover, no matter whether t_{k-1} is an S- or an N-node, there exists $u \in M(t_{k-1}) - M(t_k)$ such that u is adjacent to both a, b. Notice also that k is adjacent to u, a, b since $k \in K(t_i)$. Then, if x is adjacent to u, the vertices z, x, k, u, b would induce an $\overline{F}_1$, otherwise, the vertices z, x, a, u, b would induce a P_5.

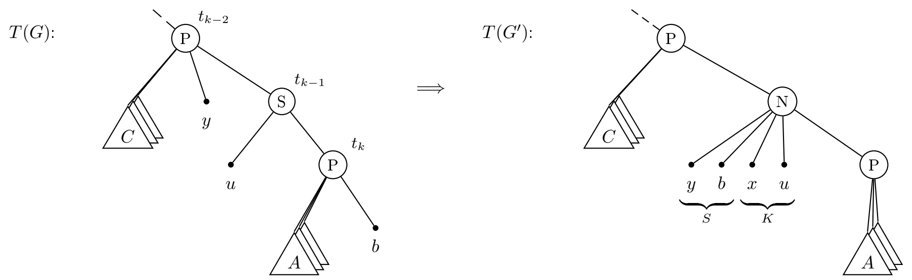
Observe that if there are no x-partly-adjacent nodes then the root of $T(G)$ is either an x-fully-adjacent node or an x-non-adjacent node. This means that $V_P = V_S = V_{N_K} = V_{N_S} = \emptyset$ and Case (i) trivially applies. Moreover the same situation holds whenever t_k is the root of the tree $T(G)$. Thus, in the following, we assume that t_k exists and t_k is not the root of $T(G)$. Now, if x sees V_S and misses V_P and t_{k-1} is not an N-node, then Case (i) applies again; note that from Properties A1 and A3, x also sees V_{N_K} and misses V_{N_S}.

Suppose next that t_k is not the root of $T(G)$ and that x sees $y \in V_P$, or misses $z \in V_S$, or t_{k-1} is an N-node, and that G' is P_4-sparse; since t_k is a P-node, we distinguish the following cases:

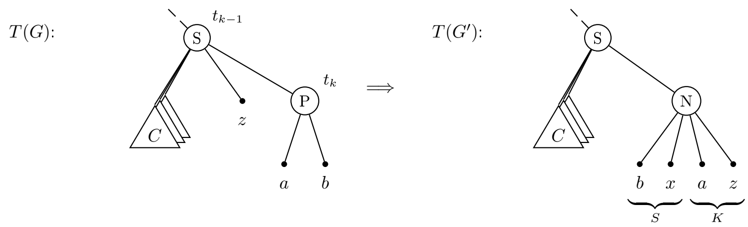
(a) *t_{k-1} is an S-node*: let $u \in M(t_{k-1}) - M(t_k)$; then u is adjacent to both a, b.



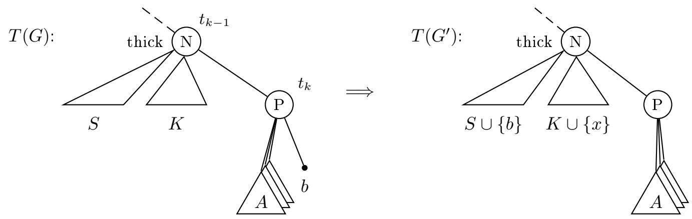
(a) In Case (i) C denotes the subtrees rooted at t_{k-1} that contain vertices of $M(t_{k-1}) - M(t_k)$ which are adjacent to x, A denotes the subtrees rooted at t_k that contain vertices of $M(t_k)$ which are adjacent to x, and B denotes the subtrees rooted at t_k that contain vertices of $M(t_k)$ which are not adjacent to x.



(b) In Case (ii) C denotes the subtrees rooted at t_{k-2} that contain vertices of $M(t_{k-2}) - \{M(t_{k-1}) \cup \{y\}\}$ which are not adjacent to x and A denotes the subtrees rooted at t_k that contain vertices of $M(t_k)$ which are adjacent to x. In $T(G')$ note that if $C = \emptyset$ then the parent of $p(x)$ becomes a child of t_{k-3} in $T(G)$ (so that t_{k-2} is removed in $T(G')$) and if A contains only one leaf-node then it becomes a child of $p(x)$.



(c) In Case (iii) C denotes the subtrees rooted at t_{k-1} that contain vertices of $M(t_{k-1}) - \{M(t_k) \cup \{z\}\}$ which are adjacent to x. In $T(G')$ note that if $C = \emptyset$ then $p(x)$ becomes a child of t_{k-2}.



(d) In Case (iv) A denotes the subtrees rooted at t_k that contain vertices of $M(t_k)$ which are adjacent to x. In $T(G')$ note that if A contains only one leaf-node then it becomes a child of $p(x)$.

Figure 7: Illustrating the four cases of Lemma 4.4 and the corresponding updates of the md-tree.

- Suppose that x sees $y \in V_P$. Since $y \in V_P$, y misses a, b, u. Then, x sees u, otherwise x, y, u, a, b

would induce a P_5. Moreover, y is a child of t_{k-2}: if y were a child of t_i, where $i < k-2$, then t_{i+1} would be an S- or an N-node and thus there would exist a vertex v such that v sees u, a, b whereas v misses y; then, the vertices x, y, u, v, b would induce an $\overline{F}_1$ or an F_2 depending on whether x is adjacent to v or not. Next, we show that u is t_{k-1}'s only child other than t_k; if not, then since t_{k-1} is an S-node, there would exist adjacent vertices $u, u' \in M(t_{k-1}) - M(t_k)$ and the vertices x, y, u, u', b would induce an $\overline{F}_1$ (recall that x sees any vertex $u \in M(t_{k-1}) - M(t_k)$). Additionally, x cannot miss two vertices $b, b' \in M(t_k)$ because then the vertices x, y, u, b, b' would induce either an F_1 or an F_2 depending on whether b, b' are adjacent or not. Finally, b is a child of t_k, otherwise the child of t_k that would be an ancestor of b would be an S- or N-node, and thus there would exist a vertex a' adjacent to b; then, the vertices x, y, b, a', u would induce an $\overline{F}_1$. Putting everything together we obtain Case (ii).

- Suppose that x misses $z \in V_S$ and misses V_P. Because $z \in V_S$, z is adjacent to a, b.

  If $z \notin M(t_{k-1}) - M(t_k)$ then because t_{k-2} is a P- or an N-node, there exists $v \in M(t_{k-2})$ such that v misses both a, b and v, z are adjacent. By Properties A2 and A3 x is adjacent to v and the vertices x, z, v, a, b induce an F_1. Thus $z \in M(t_{k-1}) - M(t_k)$.

  If z is not a child of t_{k-1} then there exists a vertex $u' \in M(t_{k-1}) - M(t_k)$ such that z, u' are non-adjacent, since t_{k-1} is an S-node. Moreover x sees u' by Property A1 and $u' \in V_S$. Furthermore u' is adjacent to both a, b. Then the vertices x, z, u', a, b induce a $\overline{P}_5$. Hence z is a child of t_{k-1}.

  Next we show that x sees and misses exactly one vertex in $M(t_k)$. Recall that t_k has children only x-fully-adjacent nodes or x-non-adjacent nodes, since t_k is the farthest away from the root x-partly-adjacent node. If x misses two vertices $b, b' \in M(t_k)$ then b' is not adjacent to a because t_k is a P-node and there would have been an x-partly-adjacent node as a child of t_k. For the same reason if x sees two vertices $a, a' \in M(t_k)$ then a', b are non-adjacent. Notice that z sees $M(t_k)$ since z is a child of t_{k-1}. In the former situation, the vertices x, z, a, b, b' would induce an F_1 or an F_2 depending on whether b, b' are adjacent or not, and in the later situation the the vertices x, z, a, a', b would induce an $\overline{F}_2$ or an $\overline{F}_1$ depending on whether a, a' are adjacent or not. Thus we obtain precisely Case (iii).

(b) t_{k-1} *is an N-node*: let $k \in K(t_{k-1})$ and $s, s' \in S(t_{k-1})$ such that k sees s but misses s'; clearly, k sees a, b whereas s, s' miss them. By Property A1, x sees the clique $K(t_{k-1})$ of the spider $G(t_{k-1})$; thus, x sees k. By Lemma 4.3, x either misses the independent set $S(t_{k-1})$ of $G(t_{k-1})$ or sees $S(t_{k-1})$. We distinguish the two cases.

- Suppose that x misses $S(t_{k-1})$. Then x misses V_{N_S}. Vertex x misses V_P as well: if it saw $y \in V_P$, then the vertices x, y, k, s, b would induce an F_1; recall that x sees k by Property A1. Additionally, x sees V_S: if it missed $z \in V_S$, then the vertices x, z, a, s, s' would induce an F_1. This is covered by Case (i).

- Suppose that x sees $S(t_{k-1})$. Then, x sees V_S: if it missed $z \in V_S$, then the vertices x, z, s', a, b would induce an $\overline{F}_2$; recall that $s' \in S(t_{k-1})$. Additionally, x misses V_P: if it saw $y \in V_P$, then the vertices x, y, s', k, b would induce an F_1. Furthermore, t_{k-1} corresponds to a thick spider with $\ell \geq 2$; if not, $\ell > 2$ and there would exist vertices $s', s'' \in S(t_{k-1})$ that would miss $k \in K(t_{k-1})$ and the vertices x, k, s', s'', b would induce an F_1. Finally, x sees all the vertices in $M(t_k)$ except for b; if x also missed $b' \in M(t_k)$ then the vertices x, k, s', b, b' would induce an F_1 ($bb' \notin E(G)$) or an F_2 ($bb' \in E(G)$). Then also observe that b is a child of t_k, since t_k does not have an x-partly-adjacent node as a child. This is precisely Case (iv).

∎

The case where t_k is an S-node is precisely the complement version of Lemma 4.4: we need to exchange P- and S-nodes, thin and thick spiders, their cliques and independent sets, and what x sees/misses in the conditions of Lemma 4.4.

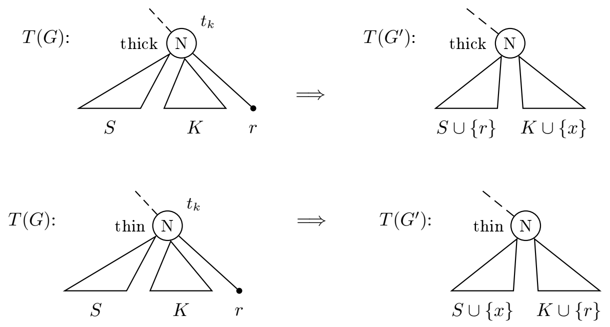


Figure 8: Illustrating cases (i) and (ii) of Lemma 4.6 and the corresponding updates of the md-tree.

Lemma 4.5. *Suppose that the x-partly-adjacent nodes of the md-tree $T(G)$ lie on a path $t_0 t_1 \cdots t_k$, where t_0 is the root of $T(G)$ and t_i is the parent of t_{i+1} for each $i = 0, 1, \ldots, k-1$. If t_k is an S-node then G' is P_4-sparse if and only if one of the following four (mutually exclusive) cases holds:*

(i) Vertex x sees V_S and V_{N_K}, and misses V_P and V_{N_S}.

(ii) Vertex x sees V_{N_K}, all but one vertex, say, y, in V_S, and misses V_P and V_{N_S} where

(ii.1) vertex y is a child of node t_{k-2} (which is an S-node),

(ii.2) node t_{k-1} is a P-node with two children, the node t_k and one vertex, say, u (which is non-adjacent to x), and

(ii.3) vertex x sees only a single vertex of $M(t_k)$, which is a child of t_k.

(iii) Vertex x sees V_S, V_{N_K}, and exactly one vertex, say, z, in V_P, and misses V_{N_S} where

(iii.1) vertex z is a child of node t_{k-1} (which is a P-node), and

(iii.2) node t_k has two children a, b, which are leaf-nodes such that a is adjacent and b is non-adjacent to x.

(iv) The node t_{k-1} is an N-node corresponding to a thin spider with clique $K(t_{k-1})$, vertex x misses V_P, V_{N_S}, $K(t_{k-1})$, and all but one vertex, say, b, in $M(t_k)$, and sees V_S and $V_{N_K} - K(t_{k-1})$.

Proof. Since the P_4-sparse graphs are complement-invariant (Lemma 2.1), we consider the graph $\overline{G}$: its md-tree $T(\overline{G})$ is identical in structure to $T(G)$ except that P-nodes have become S-nodes and vice versa, thin spiders have become thick and vice versa, and their cliques and independent sets have been swapped. Since a node in $T(\overline{G})$ is x-partly-adjacent iff its corresponding node in $T(G)$ is x-partly-adjacent, Lemma 4.4 applies and gives us necessary and sufficient conditions for $\overline{G}'$ to be P_4-sparse. By exchanging P- and S-nodes, thin and thick spiders, their cliques and independent sets, and what x sees/misses in these conditions, we obtain the conditions of the lemma, which are the necessary and sufficient conditions for G' to be P_4-sparse. ∎

Lemma 4.6. *Suppose that the x-partly-adjacent nodes of the md-tree $T(G)$ lie on a path $t_0 t_1 \cdots t_k$, where t_0 is the root of $T(G)$ and t_i is the parent of t_{i+1} for each $i = 0, 1, \ldots, k-1$. If t_k is an N-node and the partition of the spider $G(t_k)$ is (S, K, R), then G' is P_4-sparse if and only if the conditions in one of the following three (mutually exclusive) cases hold:*

(i) Vertex x sees $S \cup K$ and misses $M(r)$ where $R = \{r\}$, sees V_S and V_{N_K}, and misses V_P and V_{N_S}, the spider corresponding to t_k is a thick spider, and the node r is a leaf.

(ii) Vertex x misses $S \cup K$ and sees $M(r)$ where $R = \{r\}$, sees V_S and V_{N_K}, and misses V_P and V_{N_S}, the spider corresponding to t_k is a thin spider, and the node r is a leaf.

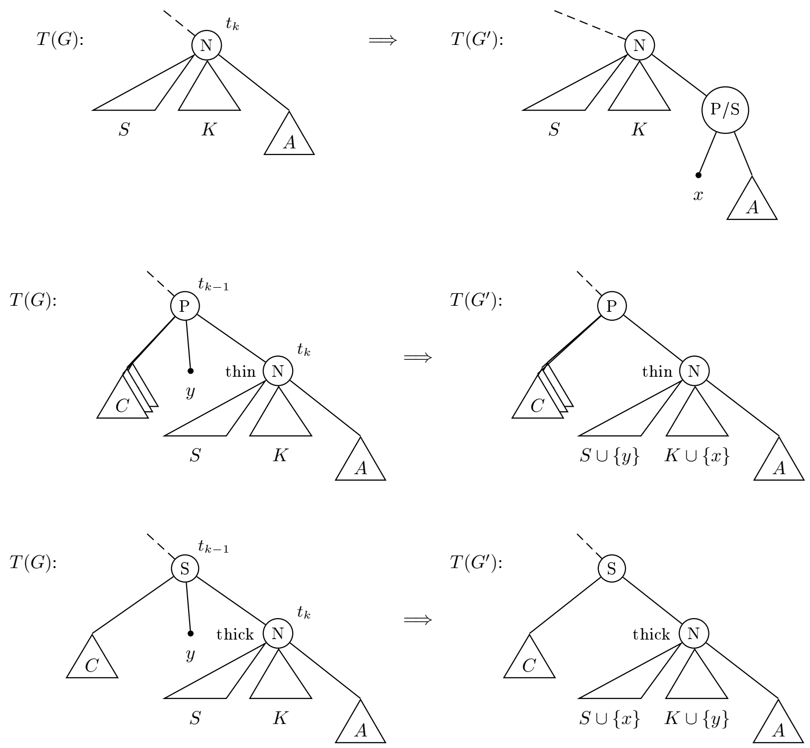


Figure 9: Illustrating cases (iii.1), (iii.2), and (iii.3) of Lemma 4.6 and the corresponding updates of the md-tree.

(iii) Vertex x sees K, misses S, and one of the following three cases holds:

- *(iii.1) vertex x sees V_S and V_{N_K}, and misses V_P and V_{N_S};*
- *(iii.2) vertex x sees V_S, V_{N_K}, and exactly one vertex, say, y, in V_P, and misses V_{N_S} where y is a child of t_{k-1}, the spider corresponding to t_k is thin, and x sees $M(r)$ (if $R = \{r\}$);*
- *(iii.3) vertex x sees V_{N_K}, all but one vertex, say, y, in V_S, and misses V_P and V_{N_S} where y is a child of t_{k-1}, the spider corresponding to t_k is thick, and x misses $M(r)$ (if $R = \{r\}$).*

Proof. For the "if"-part, we show that if one of the stated cases applies then G' admits an md-tree with the properties described in Lemma 2.2. We describe Case (iii.1). In a similar manner it is not difficult to construct $T(G')$ for the rest of the cases as we depict in Figures 8–9. If Case (iii.1) holds and $R = \emptyset$ then x becomes a child of t_k in $T(G')$; if Case (iii.1) holds and $R = \{r\}$ then x either sees or misses $M(r)$. In the former case, x becomes a child of an S-node in $T(G')$ while in the latter, x becomes a child of a P-node (Figure 9, top figure). Therefore by Lemma 2.2, G' is P_4-sparse.

Now assume that G' is P_4-sparse. We need to show that exactly one of the stated cases holds. By Property P4 there exist four vertices $a, b, a', b' \in M(t_k)$ such that $xa, xa', ab \in E(G')$ and $xb, xb', a'b' \notin E(G')$. For G' to be P_4-sparse the following properties must hold:

B1: *x must be adjacent to all but at most one vertex in V_S:* if x were non-adjacent to vertices $y, y' \in V_S$ then the vertices x, a', b', y, y' would induce in G' either an $\overline{F}_1$ or an $\overline{F}_2$ depending on whether y, y' are adjacent or not. Thus G' would not be P_4-sparse.

B2: *x must be adjacent to at most one vertex in V_P:* if x were adjacent to vertices $z, z' \in V_P$ then since both z, z' are non-adjacent to a, b, the vertices x, a, b, z, z' would induce an F_1 or an F_2 depending on whether z, z' are adjacent or not, respectively.

B3: *x must see the entire V_{N_K} and miss the entire V_{N_S}*: x sees the the entire V_{N_K} by Lemma 4.3. If x were adjacent to a vertex z belonging to V_{N_S} of a spider $G(t_i)$ associated with a node t_i $(0 \leq i \leq k-1)$ then there exists $k \in K(t_i)$ such that k is non-adjacent to z; then $k \in V_{N_K}$ and x is adjacent to k. Moreover, both b, b' are adjacent to k and non-adjacent to z, since every vertex of $G(t_k)$ sees $K(t_i) \subseteq V_{N_K}$ and misses $S(t_i) \subseteq V_{N_S}$. Then, the vertices z, x, k, b, b' would induce either an F_1 or an F_2 in G' depending on whether b, b' are adjacent or not.

Let us consider the cases that may arise. First, suppose that t_k is the root of $T(G)$ or x sees V_S and misses V_P; note that if t_k is the root of $T(G)$ then $V_S = V_P = \emptyset$. In accordance with Lemma 4.3, we distinguish the three following cases (a), (b) and (c):

(a) *x sees $S \cup K$*: Then, since t_k is the lowermost x-partly-adjacent node in $T(G)$, $R = \{r\}$ and r is an x-non-adjacent node. Thus, x misses every vertex in $M(r)$. Let $z \in M(r)$. By the definition of the spider $G(t_k)$, z sees K and misses S. The two following properties are satisfied:

- $G(t_k)$ is a thick spider: If $G(t_k)$ were a thin spider then there would exist two non-adjacent vertices $s_1, s_2 \in S$ such that both s_1 and s_2 miss a vertex $k \in K$; but then, the vertices z, k, x, s_1, s_2 would induce an F_1.
- r is a leaf: if r were not a leaf, then there would exist $z' \in M(r)$ such that $z' \neq z$. Let $s \in S$ and $k \in K$ be two non-adjacent vertices of $S \cup K$; but then, the vertices s, x, k, z, z' would induce either an F_1 or an F_2 depending on whether z, z' are adjacent or not.

The above along with Property B3 imply that this is precisely Case (i).

(b) *x misses $S \cup K$*: Then, since t_k is the lowermost x-partly-adjacent node in $T(G)$, $R = \{r\}$ and r is an x-fully-adjacent node. Thus, x sees every vertex in $M(r)$. Let $z \in M(r)$. By the definition of the spider $G(t_k)$, z sees K and misses S. The two following properties are satisfied:

- $G(t_k)$ is a thin spider: If $G(t_k)$ were a thick spider then there would exist two non-adjacent vertices $s_1, s_2 \in S$ which would both see a vertex $k \in K$; but then, the vertices x, z, k, s_1, s_2 would induce an F_1.
- r is a leaf: If r were not a leaf, then there would exist $z' \in M(r)$ such that $z' \neq z$. Let $s \in S$ and $k \in K$ be two adjacent vertices of $S \cup K$; but then, the vertices x, z, z', k, s would induce either an $\overline{F}_1$ or an $\overline{F}_2$ depending on whether z, z' are adjacent or not.

This is precisely Case (ii).

(c) *x sees K and misses S*: In light of Property B3, this is covered by Case (iii.1).

Now suppose that t_k is not the root of $T(G)$ and it is not the case that x sees V_S and misses V_P; then, due to Properties B1 and B2, x misses exactly one vertex of V_S or sees exactly one vertex of V_P. We distinguish the two following cases (d) and (e):

(d) *Suppose that x sees $y \in V_P$*: By Property B2, x cannot see another vertex of V_P. First we prove that y must be a child of t_{k-1}. Suppose that $y \in M(t_i) - M(t_{i+1})$ where $i \leq k-2$; then t_i is a P-node and consequently t_{i+1} is an S- or an N-node. Let $z \in M(t_{i+1}) - M(t_{i+2})$. Observe that y misses every vertex in $M(t_{i+1}) \supseteq M(t_k)$ whereas z sees every vertex in $M(t_k)$. If x is adjacent to z then the vertices y, x, z, b, b' induce either an F_1 or an F_2 depending on whether b, b' are adjacent or not. Hence $y \in M(t_{k-1}) - M(t_k)$. Now assume that y is not a child of t_{k-1}. Let t' be the child of t_{k-1} that is an ancestor of y. Since $t' \neq y$ and t' is an N-node or an S-node, there exists $z' \in M(t')$ such that y and z' are adjacent in G; then the vertices z', y, x, a, a' induce either an F_1 or an F_2 depending on whether a, a' are adjacent or not. Therefore y must be a child of t_{k-1}. The four following properties are satisfied:

- x sees V_S: Otherwise, if there existed $p \in V_S$ non-adjacent to x then the vertices x, y, p, b, b' would induce either an F_1 or an F_2, since p is adjacent to y and to every vertex of $M(t_k)$ as it belongs to V_S.
- x sees K and misses S: If x saw $S \cup K$ then $R = \{r\}$ and x would miss a vertex $z \in M(r)$ since t_k is an x-partly-adjacent node. Let $s \in S, k \in K$ be two non-adjacent vertices in G. Then the vertices y, x, s, k, z would induce an F_1, since y misses s, k, z; a contradiction. If x missed $S \cup K$ then $R = \{r\}$ and x would see a vertex $z \in M(r)$. Let $s \in S, k \in K$ be two adjacent vertices in G. Then the vertices y, x, z, k, s would induce a P_5; a contradiction again. Hence, by Lemma 4.3, x sees K and misses S.
- $G(t_k)$ is a thin spider. By the previous observation, we know that x sees K and misses S. If $G(t_k)$ were a thick spider then there would exist two non-adjacent vertices $s_1, s_2 \in S$ that are both adjacent to a vertex $k \in K$. But then the vertices y, x, k, s_1, s_2 would induce an F_1.
- x sees $M(r)$ (if $R = \{r\}$). By an earlier observation, we know that x sees K and misses S. Let $s \in S$ and $k \in K$ be two adjacent vertices in G. If x missed a vertex $z \in M(r)$ then the vertices y, x, k, s, z would induce an F_1 since z sees k but misses s.

Putting everything together, we obtain Case (iii.2), since t_{k-1} must be a P-node.

(e) *Suppose that x misses $y \in V_S$:* By Lemma 2.1 we know that P_4-sparse graphs are complement-invariant. Consider the complement of case (d). Then the P-nodes become S-nodes and vice versa, thin spiders become thick and vice versa, and their cliques and independent sets are being swapped. Moreover by exchanging what x sees/misses in the conditions of (d) we obtain the required conditions for G' to be P_4-sparse. This is precisely Case (iii.3).

∎

The procedure that handles the addition of vertex x finds the node t_k and takes advantage of Lemmas 4.4–4.6 to check and modify the tree $T(G)$. It starts from the leaves of the md-tree $T(G)$ which correspond to the neighbors of x and moving in a bottom-up fashion constructs the set A of internal nodes of $T(G)$ having at least one x-fully-adjacent child. Then, it splits A obtaining the set *Full* of x-fully-adjacent nodes of $T(G)$ and a subset *Partial* of the set of x-partly-adjacent nodes, from which it determines t_k (vertex t' of Step 3); in this way, this can be done in $O(d)$ time. Furthermore because in each case of Lemmas 4.4–4.6, x sees V_{N_K} and all but at most one of the elements of V_S, and the parent of a P-node cannot be a P-node, the following holds:

Observation 4.7. *For each node $t \in$ Partial at distance at least 4 from the root of the tree $T(G)$, if none of t's parent, grandparent, great-grandparent, and great-great-grandparent belongs to Partial, then the graph G' is not P_4-sparse.*

In detail, the procedure to add a vertex x works as follows:

Procedure VERTEX_ADD(vertex x)
1. $A \leftarrow \emptyset$;
   construct a queue Q whose elements are pointers to each of the leaf-nodes of $T(G)$ which correspond to the neighbors of x;
   `while` the queue Q is not empty `do`
      remove from Q an element (i.e., a pointer to a node, say, t, of $T(G)$);
      increment the *counter*-field of the parent $p(t)$ of t by 1 and let its new value be *val*;
      `if` $val = 1$
      `then` insert in A a pointer to $p(t)$;
      `if` $val =$ number of $p(t)$'s children
      `then` insert in Q a pointer to $p(t)$;   {*t is x-fully-adjacent*}

2. $Full \leftarrow$ set of pointers to each of the leaf-nodes of $T(G)$ which correspond to the neighbors of x;
   $Partial \leftarrow \emptyset$;
   `for` each element a of the set A `do`
   let t be the node of $T(G)$ pointed by a;
   `if` the value of t's *counter*-field is equal to the number of t's children
   `then` insert a in $Full$; {*t is x-fully-adjacent*}
   `else` insert a in $Partial$; {*t is x-partly-adjacent*}
   set t's *counter*-field equal to 0; {*reset the value of counter-field*}

3. $result \leftarrow true$;
   `for` each element a of the set $Partial$ `do`
   let t be the node of $T(G)$ pointed by a;
   `if` there exist t's parent, grandparent, great-grandparent, and great-great-grandparent
   `and` none is pointed by an element of $Partial$
   `then` $result \leftarrow false$;
   `goto` Step 4;
   mark t's parent, grandparent, great-grandparent, and great-great-grandparent (if they exist);
   `if` there exist two or more nodes pointed by elements of $Partial$ which are unmarked
   `then` $result \leftarrow false$;
   `else` $t' \leftarrow$ the unique node pointed by an element of $Partial$ which is unmarked;

4. `for` each element a of the set $Partial$ `do`
   let t be the node of $T(G)$ pointed by a;
   unmark t's parent, grandparent, great-grandparent, and great-great-grandparent;

5. `if` $result = false$
   `or` none of the cases of Lemmas 4.4–4.6 applies to t'
   by parsing the unique path from t' to the root of $T(G)$
   `then` output *false* (i.e., G' is not P_4-sparse); `return`;
   Appropriately modify $T(G)$ depending on the case of Lemma 4.4, 4.5, or 4.6 that applies to t';

We next discuss the correctness of the algorithm. First note that x-partly-adjacent nodes of $T(G)$ form a path by Theorem 4.2. In Step 1 we collect in A some x-partly-adjacent nodes and every x-fully-adjacent node, by incrementing appropriately the corresponding fields. In fact A contains every x-partly-adjacent node of $T(G)$ that has at least one child which is x-fully-adjacent node. Recall that an x-partly-adjacent node of $T(G)$ has at least one child that is not x-non-adjacent node. Thus the set $Partial$ obtained in Step 2 contains the set of x-partly-adjacent nodes that are included in A, whereas the set $Full$ contains every x-fully-adjacent node of $T(G)$. Node t_k which is the farthest x-partly-adjacent node away from the root has at least one child which is x-fully-adjacent node and, thus, t_k is included in the set $Partial$. Moreover if G' is P_4-sparse then the x-partly-adjacent nodes of $Partial$ lie on the path formed by the x-partly-adjacent nodes of $T(G)$. In the marking process executed in Step 3, every node of $Partial$ marks some of its ancestors, meaning that the only unmarked node (t') of $Partial$ is exactly node t_k. By Observation 4.7 it suffices to check 4 levels away from each x-partly-adjacent node of $Partial$, so that the marked nodes at the end of Step 3 form the path $t_0, \ldots, t_{k-1}$ of $T(G)$. Notice that Step 4 cleans up the marks on the nodes of the md-tree in preparation for the next modification and does not influence the correctness of the algorithm. Together with Lemmas 4.4–4.6 that are applied on node t', we conclude that the algorithm correctly handles the addition of a vertex x.

For the time complexity of the procedure, we need the following:

Observation 4.8. *Let d be the number of vertices of G which are adjacent to x. Then, the size of the set $Full$ at the end of Step 2 is less than $2d$, the size of the set $Partial$ at the end of Step 2 does not exceed d; consequently, the size of the set A at the end of Step 1 is less than $3d$.*

Proof. The bound on the size of the set *Full* follows from the fact that the number of x-fully-adjacent nodes is less than $2d$ (Observation 4.1); recall that *Full* contains pointers to precisely the x-fully-adjacent nodes. The bound on the size of *Partial* follows from the fact each node of $T(G)$ (except for the root) has exactly one parent and that the x-fully-adjacent nodes form at most d trees (Observation 4.1); each x-fully adjacent node that is a child of an x-partly-adjacent node is a root of such a tree, and *Partial* contains pointers to precisely the x-partly-adjacent nodes with at least one x-fully-adjacent child. The bound on the size of A follows from the previous bounds because the sets *Full* and *Partial* form a partition of A. ∎

Now we are ready to show our main result of this section.

Theorem 4.9. *Let G be a P_4-sparse graph. The addition of a vertex $x \notin V(G)$ adjacent to d vertices of G can be handled in $O(d)$ time.*

Proof. The fact that Procedure Vertex_Add performs the desired task follows from the correctness discussion above. We will next analyze its running time. Note that Step 1 guarantees that each x-fully-adjacent node will at some point be inserted in Q; thus, since a node is inserted at most once in Q, Observation 4.1 implies that the while loop in Step 1 is executed less than $2d$ times. Since inserting and deleting elements can be done in constant time, and we can access the parent of a node in constant time, Step 1 takes $O(d)$ time. Similarly, Step 2 takes $O(d)$ time; the size of the set A is $O(d)$ (Observation 4.8). Step 3 also takes $O(d)$ time; note that the parent, grandparent, great-grandparent, and great-great-grandparent of a node are accessed in constant time following 4 pointers to parent-nodes. Similarly, Step 4 takes $O(d)$ time. Finally, for Step 5, Observations 4.7 and 4.8 imply that the depth of node $t'(= t_k)$ does not exceed $4d$. Moreover, an exhaustive check of the cases in Lemmas 4.4–4.6 ensures that Step 5 can also be completed in $O(d)$ time: we walk from t' to the root and we check if the conditions of one of the possible cases are met by taking advantage of the parent-pointers and the fact that the value of the *counter*-field of a node is equal to the number of its children that are x-fully-adjacent; it is important to note that checking whether x sees the module $M(t)$ associated with a node t is done by looking whether t is in the set *Full*. Therefore, the overall time complexity is $O(d)$. ∎

4.2 Deleting a Vertex

Let $v \in V(G)$ be a vertex with d incident edges in G which has to be deleted. Clearly, the graph G' which results after the deletion of v is a P_4-sparse graph as it is an induced subgraph of G (see Lemma 2.1). Hence we focus on properly updating the md-tree $T(G)$ so that we obtain the md-tree $T(G')$.

Let us first consider the case where the parent $p(v)$ of v in $T(G)$ is an N-node t such that the spider partition of $G(t)$ is (S, K, R). We distinguish the following cases:

(i) $v \in S$: First suppose that $S = \{v, v'\}$, $K = \{k, k'\}$, and let v be adjacent to k: then, the spider is replaced by an S-node with children the vertex k' and a P-node; if $R = \emptyset$, then this P-node has as children the vertices v' and k, else if $R = \{r\}$, it has as children the vertex v' and an S-node with children the vertex k and the node r. Now, suppose that $|S| = |K| \geq 3$ and let $f(v) = k \in K$. If the spider is thin then: if $R = \emptyset$, then after the removal of v, k is removed from the set K and included into a new set R, so that $R = \{k\}$; if $R = \{r\}$, then k is removed from the set K and if r is an S-node then k is placed as a child of r, otherwise the place of r is taken by a new S-node with k and r as children. If the spider is thick, then after the removal of v, vertex k sees all the remaining vertices in $M(t)$; thus, the N-node t is replaced by an S-node with children the vertex k and the node t after we have removed the vertices v, k.

(ii) $v \in K$: Since the complement of a thin spider is a thick spider (and vice versa) with the clique and independent sets swapped (and if $R = \{r\}$, the P- and S-nodes in the subtree rooted at r

swapped as well), this is the complement version of the previous case and takes the same time to handle.

(iii) $R = \{v\}$: In this case, v is deleted, and we obtain a spider with $R = \emptyset$.

For the case when $p(v)$ is a P- or S-node we work as noted in [21] where the appropriate modifications on $T(G)$ can be handled in $O(d)$ time. Thus, we have:

Theorem 4.10. *The deletion of a vertex v of a P_4-sparse graph G can be handled in $O(d)$ time, where d is the degree of v in G.*

Therefore by combining Theorems 3.7, 3.8, 4.9 and 4.10 we conclude with our main theorem:

Theorem 4.11. *There is a fully dynamic algorithm for recognizing P_4-sparse graphs and maintaining their modular decomposition tree, which handles additions and deletions of vertices and edges. Edge modifications can be handled in $O(1)$ time while vertex modifications can be handled in $O(d)$ time.*

5 Concluding Remarks

We have given a fully dynamic algorithm for recognizing P_4-sparse graphs. Our algorithm handles insertions and deletions of vertices and edges and runs in $O(d)$ time per operation where d is the number of edges involved in the operation.

It has become common that if a recognition algorithm decides that a graph does not belong to the required graph class then, in addition to a message stating this, the algorithm provides an evidence of non-membership, a certificate. In our case, a certificate for a graph that is not P_4-sparse is one of the seven forbidden graphs depicted in Figure 1. With a careful but not difficult augmentation, our algorithm can be made to provide such a certificate whenever an operation results in a non-P_4-sparse graph. In particular, by using the described data structure we can provide one of the forbidden subgraphs: (i) in $O(d_u + d_v)$ time for an edge modification where d_u and d_v are the degrees of the vertices involved in the edge modification and (ii) in $O(n)$ time for vertex addition. Providing certificate within an optimal running time is an interesting open problem; a possible approach might involve the maintenance of an additional $O(n)$-space representation of a vertex ordering (e.g., the *factorizing permutation* [6, 24]).

The vertex addition part of our algorithm implies an incremental algorithm to construct the modular decomposition tree of a P_4-sparse graph that takes $O(d)$ time per vertex. It would be interesting to have such an algorithm for general graphs. The currently best incremental algorithm for constructing the modular decomposition tree for general graphs handles each vertex insertion in $O(n)$ time [19]. It should be noted that several linear-time algorithms are known for building the modular decomposition tree [4, 8, 18, 24]; however, none seems to be easily modified for updating the tree in an online fashion within a running time bounded by the vertex degree.

Acknowledgements

The authors would like to express their sincere thanks to the anonymous referees whose valuable suggestions helped improve the presentation of the paper.

References

[1] A. Berry, P. Heggernes, and Y. Villanger, A vertex incremental approach for maintaining chordality, *Discrete Mathematics* **306** (2006) 318–336.

[2] A. Brandstädt, V.B. Le, and J. Spinrad, *Graph Classes – a Survey*, SIAM Monographs in Discrete Mathematics and Applications, SIAM, Philadelphia, 1999.

[3] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14** (1985) 926–984.

[4] A. Cournier and M. Habib, A new linear algorithm for modular decomposition, *Proc. 19th Int'l Colloquium on Trees in Algebra and Programming (CAAP'94)*, LNCS **787** (1994) 68–84.

[5] C. Crespelle, Fully dynamic representations of interval graphs, *Proc. 35th Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009)*, LNCS **5911** (2009) 77–87.

[6] C. Crespelle and C. Paul, Fully-dynamic recognition algorithm and certificate for directed cographs, *Discrete Appl. Math.* **154** (2006) 1722–1741.

[7] C. Crespelle and C. Paul, Fully dynamic algorithm for recognition and modular decomposition of permutation graphs, *Algorithmica*, **58** (2010) 405–432.

[8] E. Dalhaus, J. Gustedt, and R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, *J. Algorithms* **41** (2001) 360–387.

[9] X. Deng, P. Hell, and J. Huang, Linear time representation algorithms for proper circular arc graphs and proper interval graphs, *SIAM J. Comput.* **25** (1996) 390–403.

[10] V. Giakoumakis and J.-M. Vanherpe, On extended P_4-reducible and P_4-sparse graphs, *Theoret. Comput. Sci.* **180** (1997) 269–286.

[11] E. Gioan and C. Paul, Dynamic dstance hereditary graphs using split decomposition, *Proc. 18th Int'l Symposium on Algorithms and Computation (ISAAC 2007)*, LNCS **4825** (2007) 41–51.

[12] P. Heggernes and F. Mancini, Dynamically maintaining split graphs, *Discrete Appl. Math.* **157** (2009) 2057–2069.

[13] P. Hell, R. Shamir, and R. Sharan, A fully dynamic algorithm for recognizing and representing proper interval graphs, *SIAM J. Comput.* **31** (2002) 289–305.

[14] C. Hoàng, Perfect graphs, Ph.D. Thesis, McGill University, Montreal, Canada, 1985.

[15] L. Ibarra, Fully dynamic algorithms for chordal graphs and split graphs, *ACM Transactions on Algorithms*, **4** (2008) Article 40.

[16] B. Jamison and S. Olariu, Recognizing P_4-sparse graphs in linear time, *SIAM J. Comput.* **21** (1992) 381–406.

[17] B. Jamison and S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Appl. Math.* **35** (1992) 115–129.

[18] R.M. McConnell and J. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* **201** (1999) 189–241.

[19] J.H. Muller and J. Spinrad, Incremental modular decomposition, *J. ACM* **36** (1989) 1–19.

[20] S.D.Nikolopoulos, L. Palios, and C. Papadopoulos, A fully dynamic algorithm for the recognition of P_4-sparse graphs, *Proc. 32nd Workshop on Graph-Theoretic Concepts in Computer Science (WG 2006)*, LNCS **4271** (2006) 256–268.

[21] R. Shamir and R. Sharan, A fully dynamic algorithm for modular decomposition and recognition of cographs, *Discrete Appl. Math.* **136** (2004) 329–340.

[22] J. Spinrad, P_4-trees and substitution decomposition, *Discrete Appl. Math.* **39** (1992) 263–291.

[23] M. Tedder and D. Corneil, An optimal edges-only fully dynamic algorithm for distance-hereditary graphs, *Proc. 24th Int'l Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, LNCS **4393** (2007) 344–355.

[24] M. Tedder, D. Corneil, M. Habib, and C. Paul, Simpler linear-time modular decomposition via recursive factorizing permutations, *Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, LNCS **5125** (2008) 634–645.