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

Stavros D. Nikolopoulos[1], Leonidas Palios[1], and Charis Papadopoulos[2,⋆]

[1] Department of Computer Science, University of Ioannina
P.O. Box 1186, GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr
[2] Department of Informatics, University of Bergen
P.B. 7800, N-5020 Bergen, Norway
charis@ii.uib.no

Abstract. We consider 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 conditions 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. Moreover, by maintaining the children of each node of the modular decomposition tree in a binomial heap, we can handle vertex deletions in $O(\log n)$ time, at the expense of needing $O(\log n)$ time for each edge modification and $O(d \log n)$ time for the addition of a vertex adjacent to d vertices.

Keywords: fully dynamic algorithms, P_4-sparse graphs, 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 yields 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 modifications 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

⋆ Work by Charis Papadopoulos was carried while he was a graduate student at the Department of Computer Science, University of Ioannina.

F.V. Fomin (Ed.): WG 2006, LNCS 4271, pp. 256–268, 2006.

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 recognition algorithms have been proposed by Hsu [12] for interval graphs and by Deng *et al.* [8] for connected proper interval graphs. Ibarra [13] has given an edges-only fully dynamic algorithm for chordal graphs which handles each edge operation in $O(n)$ time and an edges-only fully dynamic algorithm for split graphs which handles each edge operation in $O(1)$ time. More recently, Hell *et al.* [10] 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; Shamir and Sharan [19] 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; finally, Crespelle and Paul have presented fully dynamic algorithms for directed cographs [5] and permutation graphs [6] which require $O(d)$ time if d arcs are involved, and $O(n)$ time, respectively. 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 [15] which handles the insertion of a vertex of degree d in $O(d)$ time.

Researchers have also considered the problem of the dynamic maintenance of the modular decomposition tree of a graph: Muller and Spinrad [18] have given an incremental algorithm, which handles each vertex insertion in $O(n)$ time; for cographs, Corneil *et al.* [3] have given an optimal incremental algorithm, 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*; these are the graphs in which every set of five vertices induces at most one chordless path on four vertices [11]. They are perfect and also perfectly orderable [11], and properly contain the cographs, the P_4-reducible graphs, etc. (see [1,15,16]). 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 on graphs featuring "local density"; local density is often associated with the absence of P_4s and the P_4-sparse graphs are unlikely to have many 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. Moreover, in order to improve the time complexity of the vertex deletion operation, we can maintain the children of each node of the modular decomposition tree in a binomial heap [2]. Then, we can handle vertex deletions in $O(\log n)$ time; the drawback is that then the time required for each edge modification becomes $O(\log n)$ and for the addition of a vertex adjacent to d vertices becomes $O(d \log n)$.

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.

Modular Decomposition and P_4-sparse Graphs. A subset M of vertices of a graph G is a *module* of G, if every vertex outside M is either adjacent to all vertices in M or to none of them. The emptyset, 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 1). 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 [4,7,17].

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 *p*arallel module) if the subgraph $G[M(t)]$ is disconnected, S (for *s*eries module) if the complement of $G[M(t)]$ is disconnected, or N (for *n*eighborhood 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

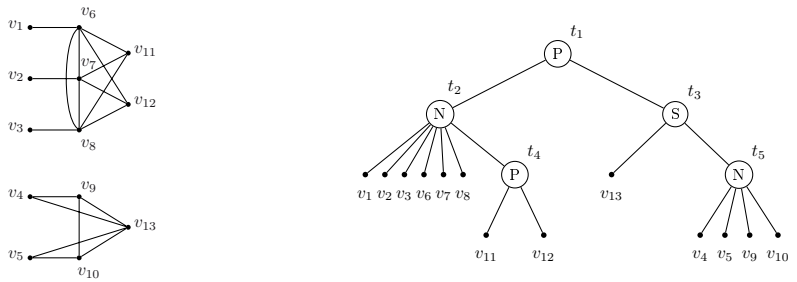

Fig. 1. A disconnected P_4-sparse graph on 13 vertices and its md-tree

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 holds:

- if t is a P-node, $G(t)$ is an edgeless graph;
- if t is an S-node, $G(t)$ is 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 [9] showed that:

Lemma 1. *Let $T(G)$ be the modular decomposition tree of a graph G. Then, 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 set, and the set K is a clique;
C2: each vertex in R is adjacent to all the vertices in K and to no vertex in S;
C3: there exists a bijection $f : S \longrightarrow K$ such that for each vertex $v \in S$, either (i) $N(v) \cap K = \{f(v)\}$ or (ii) $N(v) \cap K = K - \{f(v)\}$.

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.

3 The Fully-Dynamic Algorithm

As mentioned, our algorithm maintains the modular decomposition tree $T(G)$ of the 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 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 $|M(t_u)| = 1$ ($|M(t_v)| = 1$, resp.) then $t_u = u$ ($t_v = v$, resp.). Without loss of generality, we make the following assumption:

Assumption 1. *We assume that $|M(t_v)| \geq |M(t_u)|$.*

Then, the following 3 lemmata cover all possible cases that may arise.

Lemma 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 $|M(t_u)| = |M(t_v)| = 2$.*

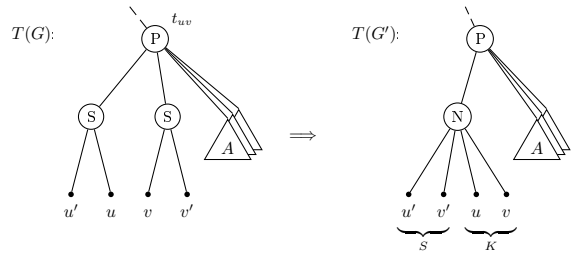

Fig. 2. Illustrating Lemma 2 and the corresponding updates of the md-tree

Lemma 3. *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 $M(t_v)$;

(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 every vertex 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 vertices of $M(r)$.

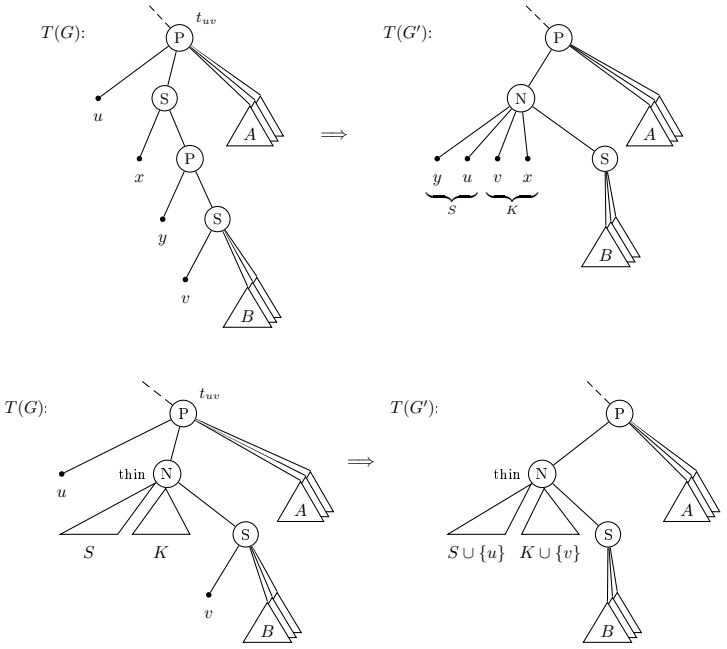

Fig. 3. Illustrating cases (ii) and (iii) of Lemma 3

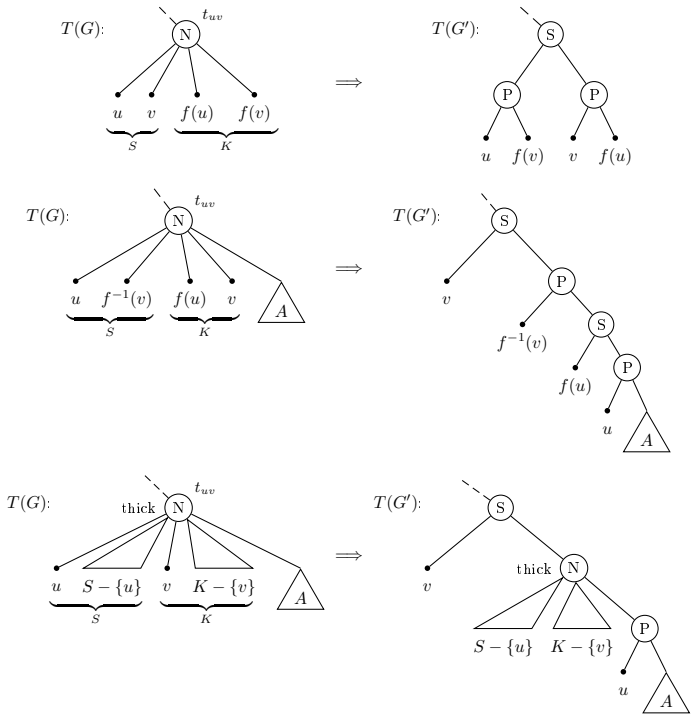

Fig. 4. Illustrating the cases of Lemma 4

Lemma 4. *Let $|M(t_u)| = 1$ (i.e., $M(t_u) = \{u\}$) and suppose that t_{uv} is an N-node such that (S, K, R) is 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.*

3.2 Removing an Edge

Since the P_4-sparse graphs have the complement-invariant property, we take advantage of the following theorem by Shamir and Sharan [19]:

Theorem 1. [19] *Let Π be a complement-invariant graph property. Let Alg be a dynamic algorithm for Π-recognition, which supports either edge additions only or edge deletions only, and is based on modular decomposition. Then Alg can be extended to support both operations with the same time complexity.*

3.3 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 P_4-sparse, 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 $T(G)$ 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. Because the x-fully-adjacent nodes form a forest of subtrees of $T(G)$ whose total number of leaves is d and because every internal node in $T(G)$ (and in these subtrees) has at least two children, we have:

Observation 1. *The number of x-fully-adjacent (internal and leaf) nodes of $T(G)$ is less than $2d$, where d is the number of vertices of G adjacent to x.*

In turn, for the x-partly-adjacent nodes, the fact that the module of an S-node induces a connected graph, the module of a P-node induces a graph whose complement is connected, and the module of an N-node induces a graph which is connected and whose complement is also connected implies:

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.

Additionally, the following very important property holds:

Theorem 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.*

Let $\rho_x = t_0 t_1 \cdots t_k$ denote the path in $T(G)$ containing all the x-partly-adjacent nodes (Theorem 2) where t_0 is the root of $T(G)$ and t_k is the x-partly-adjacent node farthest away from the root. Then, Theorem 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. Additionally, for the x-partly-adjacent N-nodes, the following holds:

Lemma 5. *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.*

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

$$V_P = \bigcup_{\substack{t_i \text{ is a P-node} \\ 0 \le i < k}} (M(t_i) - M(t_{i+1})), \qquad V_{N_S} = \bigcup_{\substack{t_i \text{ is an N-node} \\ 0 \le i < k}} S(t_i),$$

$$V_S = \bigcup_{\substack{t_i \text{ is an S-node} \\ 0 \le i < k}} (M(t_i) - M(t_{i+1})), \qquad V_{N_K} = \bigcup_{\substack{t_i \text{ is an N-node} \\ 0 \le i < k}} K(t_i),$$

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 lemmata:

Lemma 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 $T(G)$. 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 and 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, in $M(t_k)$, and misses V_P and $V_{N_S} - S(t_{k-1})$.*

The case where t_k is an S-node is precisely the complement version of Lemma 6: 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 6.

Lemma 7. *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 $T(G)$. 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:*

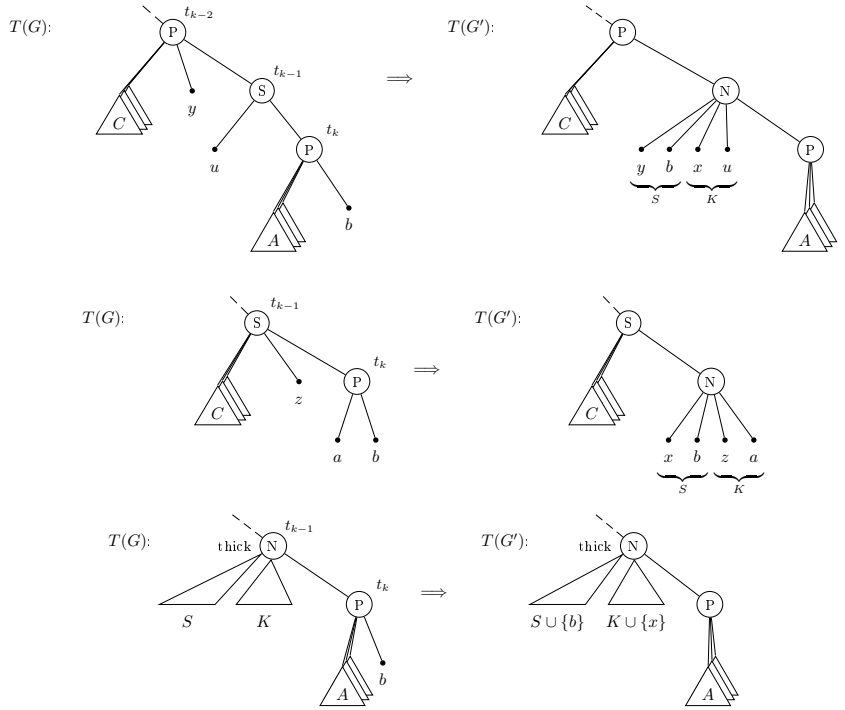

Fig. 5. Illustrating cases (ii), (iii), and (iv) of Lemma 6

(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 $V_{N_K} - K(t_{k-1})$.

Lemma 8. *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 $T(G)$. 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 one of the following three (mutually exclusive) cases holds:*

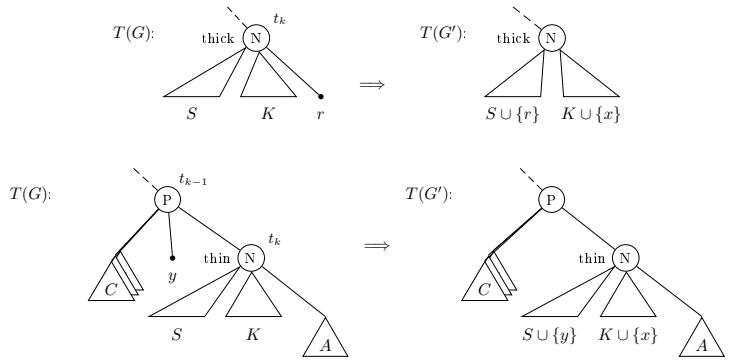

Fig. 6. Illustrating cases (i) and (ii.2) of Lemma 8

(i) Vertex x sees $S \cup K$ (and misses $M(r)$ where $R = \{r\}$): vertex x 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 sees K (and misses S): one of the following three cases holds:

- *(ii.1) vertex x sees V_S and V_{K_N}, and misses V_P and V_{N_S};*
- *(ii.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 all the vertices in $M(r)$ (if $R = \{r\}$) see x;*
- *(ii.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 all the verticts in $M(r)$ (if $R = \{r\}$) miss x.*

(iii) Vertex x misses $S \cup K$ (and sees $M(r)$ where $R = \{r\}$): vertex x 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.

Since in each case of Lemmata 6–8, x sees V_{N_K} and all but at most one of the elements of V_S, (i.e., all the x-partly-adjacent N-nodes and all but at most one x-partly-adjacent S-nodes belong to *Partial*), and since the parent of a P-node cannot be a P-node, we can show the following:

Observation 2. *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.*

This implies that for G' to be P_4-sparse, node t_k of $T(G)$ is at depth at most $4d$.

The procedure that handles the addition of vertex x finds the node t_k and takes advantage of Lemmata 6–8. 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 x-partly-adjacent nodes, from which it determines t_k (vertex t' of Step 3). In detail, the procedure 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 by 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 to 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. $t' \leftarrow$ a node of largest depth in $T(G)$ among the nodes pointed to by the elements in $Partial$;
   `if` the depth of t' in $T(G)$ exceeds $4d$
   `or` there exists a node in $T(G)$ pointed to by an element in $Partial$ which is not an ancestor of t'
   `or` none of the cases of Lemmata 6, 7, and 8 applies to t'
   `then` output *false* (i.e., G' is not P_4-sparse); `return`;
   modify $T(G)$ depending on the case of Lemma 6, 7, or 8 which applies to t';

The correctness of the algorithm follows from Theorem 2, Observation 2, and from the following facts: (i) the set of nodes of the tree $T(G)$ pointed to by the elements of the set $Full$ is precisely the set of x-fully-adjacent nodes; (ii) the set of nodes of the tree $T(G)$ pointed to by the elements of the set $Partial$ are the x-partly-adjacent nodes of $T(G)$ with at least one x-fully-adjacent child (note that $t_k \in Partial$); (iii) the node t' found in Step 3 is precisely the x-partly-adjacent node t_k farthest away from the root.

3.4 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. 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-node $p(v)$ of v in $T(G)$ is an N-node t such that the spider partition of $G(t)$ is (S, K, R). We have:

(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 K and is linked at the pointer for R; if $R = \{r\}$, then k is removed from K and if r is an S-node then k is linked as a child, otherwise the place of r is taken by an 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 $R = \emptyset$.

Next, we consider the case where the parent-node $p(v)$ of v in $T(G)$ is a P- or S-node; if $p(v)$ has more than 2 children, it suffices to simply delete v. However, caution is needed if $p(v)$ has only two children, in which case the sibling u of v needs to be linked to the grandparent $p(p(v))$ of v; furthermore, if u and $p(p(v))$ are both P- or S-nodes, then the children of u are placed as children of $p(p(v))$. Finally, in either of the remaining two cases when the parent-node $p(v)$ has 2 children, i.e., if the sibling u of v is an N-node or if the grandparent $p(p(v))$ is an N-node, then u is linked as a child of $p(p(v))$.

3.5 Time Complexity

Lemmata 2–8 and Figures 2–6 show that handling each modification requires only local checks and changes. In order to perform them efficiently, we store in each node of the md-tree $T(G)$ its type (P, S, or N), the number of its children, as well as ways to access its parent and its children, and an auxiliary field *counter* (initialized to 0). If each P- or S-node stores pointers to its parent and to a list of its children, then edge additions (and deletions, by Theorem 1) are handled in $O(1)$ time, whereas vertex additions/deletions are handled in $O(d)$ time. Alternatively, the children of a P- or S-node may be stored in a binomial min-heap [2] in which the pointer to the parent of these children in $T(G)$ is stored at the minimum element of the heap. Then, edge additions (and deletions) and vertex deletions take $O(\log n)$ time, whereas vertex additions take $O(d \log n)$ time. Our results are summarized in the following theorem.

Theorem 3. *We have described a fully dynamic algorithm for recognizing P_4-sparse graphs and maintaining their modular decomposition tree. Edge modifications can be handled in $O(1)$ time while vertex modifications can be handled in $O(d)$ time; alternatively, edge modifications and vertex deletions can be handled in $O(\log n)$ time and vertex additions in $O(d \log n)$ time.*

References

1. A. Brandstädt, V.B. Le, and J. Spinrad, *Graph Classes – a Survey*, SIAM Monographs in Discrete Mathematics and Applications, SIAM, Philadelphia, 1999.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.
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 and C. Paul, Fully-dynamic recognition algorithm and certificate for directed cographs, *Proc. 30th Int'l Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, LNCS **3353** (2004) 93–104.
6. C. Crespelle and C. Paul, Fully-dynamic algorithm for modular decomposition and recognition of permutation graphs, *Proc. 31st Int'l Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, LNCS **3787** (2005) 38–48.
7. E. Dalhaus, J. Gustedt, and R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, *J. Algorithms* **41** (2001) 360–387.
8. 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.
9. V. Giakoumakis and J.-M. Vanherpe, On extended P_4-reducible and P_4-sparse graphs, *Theoret. Comput. Sci.* **180** (1997) 269–286.
10. 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.
11. C. Hoàng, Perfect graphs, Ph.D. Thesis, McGill University, Montreal, Canada, 1985.
12. W.-L. Hsu, On-line recognition of interval graphs in $O(m + n \log n)$ time, *Combinatorics and Computer Science* 1995, LNCS **1120** (1996) 27–38.
13. L. Ibarra, Fully dynamic algorithms for chordal graphs, *Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'99)*, (1999) 923–924.
14. L. Ibarra, A fully dynamic algorithm for recognizing interval graphs using the clique-separator graph, Technical Report, DCS-263-IR, University of Victoria, 2001.
15. B. Jamison and S. Olariu, Recognizing P_4-sparse graphs in linear time, *SIAM J. Comput.* **21** (1992) 381–406.
16. B. Jamison and S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Appl. Math.* **35** (1992) 115–129.
17. R.M. McConnell and J. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* **201** (1999) 189–241.
18. J.H. Muller and J. Spinrad, Incremental modular decomposition, *J. ACM* **36** (1989) 1–19.
19. R. Shamir and R. Sharan, A fully dynamic algorithm for modular decomposition and recognition of cographs, *Discrete Appl. Math.* **136** (2004) 329–340.
20. J. Spinrad, P_4-trees and substitution decomposition, *Discrete Appl. Math.* **39** (1992) 263–291.