Fully dynamic algorithm for recognition and modular decomposition of permutation graphs

Christophe Crespelle Christophe Paul

CNRS - *Département Informatique*, LIRMM, Montpellier
{crespell,paul}@lirmm.fr

Abstract

This paper considers the problem of maintaining a compact representation ($O(n)$ space) of permutation graphs under vertex and edge modifications (insertion or deletion). That representation allows us to answer adjacency queries in $O(1)$ time. The approach is based on a fully dynamic modular decomposition algorithm for permutation graphs that works in $O(n)$ time per edge and vertex modification. We thereby obtain a fully dynamic algorithm for the recognition of permutation graphs.

1 Introduction

The *dynamic recognition and representation problem* (see e.g. [10]) for a family $\mathcal{F}$ of graphs aims to maintain a characteristic representation of dynamically changing graphs as long as the modified graph belongs to $\mathcal{F}$. The input of the problem is a graph $G \in \mathcal{F}$ with its representation and a series of modifications. Any modification is of the following: inserting or deleting a vertex (along with the edges incident to it), inserting or deleting an edge. Several authors have considered the dynamic recognition and representation problem for various graph families. [8] devised a fully dynamic recognition algorithm for chordal graphs which handles edge operations in $O(n)$ time. For proper interval graphs [7], each update can be supported in $O(d + \log n)$ time where d is the number of edges involved in the operation. Cographs, a subfamily of permutation graphs, have been considered in [10] where any modification (edge or vertex) is supported in $O(d)$ time, where d is the number of edges involved in the modification. This latter result has recently been generalised to directed cographs in [3].

This paper deals with the family of permutation graphs. Our algorithm maintains an $O(n)$ space canonical representation based on modular decomposition which enables us to answer adjacency queries in $O(1)$ time. It should be noted that in [9] a purely incremental algorithm is presented for computing the modular decomposition tree of any graph. It runs in $O(n)$ time per vertex insertion. Unfortunately, it is based on a partial representation of the graph compromising the possibility of any vertex deletion. Therefore such an algorithm cannot be applied for efficient fully dynamic recognition of permutation graphs. Our algorithm also performs in $O(n)$ time per operation, but supports insertion as well as deletion of vertices and edges. Let us note that a modification of the input graph may lead to $O(n)$ changes in the modular decomposition tree. Therefore our algorithm does not present any complexity extra cost in the maintain of the modular decomposition tree.

2 Preliminaries

2.1 Modular decomposition

Theory of modular decomposition of graphs has been widely developed since Gallai first introduced it in [5]. Here, we give some known definitions and results that we use in the following. Let $G = (V, E)$ be a graph. The neighbourhood of a vertex $x \in V$ is denoted $N(x)$ and its non-neighbourhood $\overline{N}(x)$. A subset $S \subsetneq V$ of vertices is *uniform* w.r.t. to vertex $x \in V \setminus S$ if $S \subseteq N(x)$ or $S \subseteq \overline{N}(x)$ (otherwise S is *mixed*). A *module* of a graph $G = (V, E)$ is a subset of vertices $M \subseteq V$ which is uniform w.r.t. any vertex $x \in V \setminus M$. It also follows from definition that V and $\{x\}, x \in V$ are modules of G, namely the *trivial modules*. A graph is *prime* if all its modules are trivial. A module M is *strong* if it does not overlap any module M', that is $M \cap M' = \emptyset$ or $M \subseteq M'$ or $M' \subseteq M$. Therefore, the inclusion order of the strong modules of a graph defines a tree, called the *modular decomposition tree* T. The leaves of T correspond to the singleton vertex sets of G (L_x stands for $\{x\}$) and its root is the whole vertex set of G. In the following, a node p of the modular decomposition tree could be identified with the strong module $P = V(p)$ it represents. Denoting T_p the subtree of T rooted at p, P is the set of leaves of T_p. $\mathcal{C}(p)$ is the set of children in T of p.

Thanks to the well-known modular decomposition theorem (see [1] for references), any non-leaf node p of the modular decomposition tree is labelled as follows: *parallel* if $G[P]$ is not connected; *series* if $\overline{G}[P]$ is not connected; and *prime* otherwise (the three cases are disjoint). The label of node p is denoted $label(p)$. The series and parallel nodes are also called *degenerate*

nodes. We call *maximal strong modules* of a graph $G = (V, E)$ the strong modules of G maximal wrt. inclusion and distinct from V. It is well known that the children $p_1 \dots p_k$ of p (i.e. the maximal strong modules of $G[P]$) are respectively in the parallel case, the connected components of $G[P]$, in the series case the co-connected components of G (i.e. the connected components of $\overline{G}[P]$) and in the prime case, the maximal modules of $G[P]$ distinct from P. Given a graph G, we denote $\mathcal{MSM}(G)$ the set of maximal strong modules of G.

Given a set $\mathcal{F}$ of disjoint modules, let $F \subseteq V$ be a set of vertices such that for any $M \in \mathcal{F}$, $|F \cap M| = 1$. The *quotient graph* $G/\mathcal{F}$ is the subgraph induced by the vertices of $(V \setminus \cup_{M \in \mathcal{F}} M) \cup F$. From the modular decomposition theorem, the quotient $G/\mathcal{MSM}(G)$ of G by the set of its maximal strong modules is either a stable (parallel case) or a clique (series case) or a prime graph. If with each prime node p of the modular decomposition tree T, we associate a representation of the quotient $G[P]/\mathcal{MSM}(G[P])$, then adjacency queries between any pair of vertices x, y can be answered by a search in T and in the quotient graphs.

2.2 Permutation graphs

If π is a linear order on the vertices, $\pi(x)$ denotes the rank of vertex x in π while $\pi^{-1}(i)$ is the vertex at rank i. Permutation graphs are those graphs for which there exists a pair (π_1, π_2) of linear order on the vertex set such that x and y are adjacent iff $\pi_1(x) < \pi_1(y)$ and $\pi_2(y) < \pi_2(x)$. For a graph G, such a pair $\mathcal{R} = (\pi_1, \pi_2)$ is a *realiser* of G. If $\overline{\pi}_2$ denotes the reverse order of π_1, then $\overline{\mathcal{R}} = (\pi_1, \overline{\pi}_2)$ is a realiser of $\overline{G}$. It is known that, if G is a prime graph, then its realiser is unique up to reversal and exchange[1] (the reader should refer to [6, 1] for more details on permutation graphs). Moreover, a graph G is a permutation graph iff the quotient graphs associated with the prime nodes of its modular decomposition tree are permutation graphs. It follows that associating the modular decomposition tree T with the realiser of each of its prime nodes provides an $O(n)$ space canonical representation of a permutation graph G, called hereafter the *full modular representation* of G.

Since the full modular representation contains a realiser for each prime node of T, it is well known that a realiser of the whole graph G can be retrieved in $O(n)$ time by a simple search of T. As our dynamic algorithm works in $O(n)$ time per operation, a realiser of G can be maintained without any extra cost. That guarantees the possibility of answering at any time

[1] that is (π_2, π_1), $(\overline{\pi}_1, \overline{\pi_2})$ and $(\overline{\pi}_2, \overline{\pi_1})$ are considered as the same realiser as (π_1, π_2).

adjacency queries in $O(1)$ time.

An *interval* of a linear order π on V is a set of consecutive elements of V in π. Given a pair (π_1, π_2) of linear orders, a *common interval* is a set I that is an interval of π_1 and of π_2. Recently, [11] proposed an $O(n + K)$ algorithm computing all common intervals of a pair of linear orders, K being the number of common intervals. A common interval is *strong* if it does not overlap any other common interval. Clearly common intervals of a realiser $\mathcal{R} = (\pi_1, \pi_2)$ of a permutation graph G are modules of G. The converse is false, but:

Proposition 1 *[4] The strong modules of a permutation graph $G = (V, E)$ are exactly the strong common intervals of any of its realiser $\mathcal{R}$.*

2.3 Dynamic arc operations

Unfortunately an edge modification may imply $O(n)$ changes in the modular decomposition tree. As we propose an $O(n)$ time algorithm for the vertex insertion and for the vertex deletion operations, inserting or deleting an edge e incident to vertex x will be handled by first removing x and then re-inserting x with the updated neighbourhood.

3 Vertex deletion

Let $G' = G - x$ be the graph resulting from the deletion of a vertex x in the graph G. Since the family of permutation graphs is hereditary, removing x reduces to compute the full modular representation of G' from the one of G. We shall distinguish the case where p, the parent node of x in T, is a prime node from the case where p is a degenerate node.

3.0.1 Degenerate case.

This is the easy case to handle. If x has at least two siblings, then the leaf L_x is removed from T. Assume x has only one sibling say q_2. If q_2 is a leaf, L_x and p are removed from T and q_2 becomes a child of q_1 replacing p (i.e. if q_1 is a prime node, then q_2 takes the place of p in the associated realiser). Assume q_2 is a non-leaf node. If q_1 and q_2 are both series nodes or both parallel nodes, then L_x, p and q_2 are removed from T and the children of q_2 are made children of q_1. Otherwise L_x and p are removed from T and q_2 becomes a child of q_1 replacing p. Such an update of the full modular representation can be done in $O(|\mathcal{C}(q_2)|) = O(n)$ since it leaves unchanged the quotient graphs of the prime nodes.

3.0.2 Prime case.

The removal of x may create some modules in $G'[P']$ (where $P' = P \setminus \{x\}$). We show it can be tested in $O(n)$ time. Moreover if $G'[P']$ is not a prime graph, the updated full modular representation can be computed within the same complexity.

Lemma 1 *Let $G = (V, E)$ be a prime permutation graph and x be a vertex. The non trivial strong modules of $G' = G - x$ can be partitioned in two families (possibly empty) totally ordered by inclusion.*

This is a consequence of Proposition 1 which implies that if $\mathcal{R} = (\pi_1, \pi_2)$ is the realiser of G, then for any strong module M' of G', $I = M' \cup \{x\}$ is an interval of π_1 or π_2. Therefore G' contains $O(n)$ strong modules. Moreover, as there is at most two non-trivial maximal strong modules, the root of the modular decomposition tree T' of G' has at most two non-leaf children, and each internal nodes of T' have at most one non-leaf child. The next lemma complete the information about degenerate internal nodes of T'.

Lemma 2 *Let $G = (V, E)$ be a prime permutation graph and x be a vertex. Every degenerate node of the modular decomposition tree of $G' = G - x$ has at most two children which are leaves.*

It follows that the number of modules (not necessarily strong) of G' is $O(n)$ so there is also $O(n)$ common intervals of the realiser $\mathcal{R}'$ of G'. Therefore applying [11]'s algorithm will cost $O(n)$ time to find the common intervals of $\mathcal{R}'$. From that algorithm the two families of strong common intervals (or equivalently modules) can be retrieved in $O(n)$ time. Moreover from Lemma 2 given a common interval it is possible to find its label (series, parallel or prime) in $O(1)$ time. As the realiser of each prime node of T' can be easily extracted from $\mathcal{R}$, the full modular representation of G' can be computed in $O(n)$.

Theorem 1 *Updating the full modular representation of a permutation graph under vertex deletion costs $O(n)$ time.*

4 Vertex insertion

Given a graph $G = (V, E)$, a vertex $x \notin V$ and a subset $N(x) \subseteq V$, let us define $G' = G + x$ as the graph on vertex set $V \cup \{x\}$ with edge set $E \cup \{\{x, y\} \mid y \in N(x)\})$. Each node p of the modular decomposition tree

T of G is assigned a type w.r.t. x : *linked* (resp. *notlinked*) if $P = V(p)$ is uniform w.r.t. x and $P \subseteq N(x)$ (resp. $P \subseteq \overline{N}(x)$), and *mixed* otherwise. $\mathcal{C}_l(p)$ (resp. $\mathcal{C}_{nl}(p)$) stands for the set of children of p which are typed *linked* (resp. *notlinked*) and $\mathcal{C}_m(p)$ for the set of children of p which are typed *mixed*. For $t \in \{m, l, nl\}$, we denote $F_t(p) = \bigcup_{f \in \mathcal{C}_t(p)} V(f)$.

4.1 Modular decomposition tree of $G + x$

4.1.1 Insertion node.

To compute the modular decomposition tree T' of $G' = G + x$, we can restrict our attention to a subtree T_q of T rooted at a certain node q, called the *insertion node*. q is such that T_q contains all the modifications implied by the insertion of x. Moreover, in T', x will be inserted as a child or a grand-child of node q' representing set $Q' = Q \cup \{x\}$. The discussion bellow gives the definition of q and shows that inserting x in G reduces to insert x in $G[Q]$.

Definition 1 *A node p of T is a* proper *node iff either p is uniform wrt. x, or p is a mixed node with a unique mixed child f such that $F \cup \{x\}$ is a module of $G'[P \cup \{x\}]$. Otherwise p is a* non-proper *node.*

From Definition 1, any mixed node p has at least one non-proper descendant. Indeed p always enjoys a mixed descendant having only uniform children. It follows that if any node of T is proper, then the vertex set is uniform w.r.t. x. That is x is either a universal vertex or an isolated vertex. Therefore inserting x preserves the property of being a permutation graph and the full modular representation is easy to update. That case will not be considered anymore in the following.

Definition 2 *The insertion node q is the lca of non-proper nodes of T.*

Lemma 3 *The insertion node q is such that $Q' = Q \cup \{x\}$ is a strong module of $G' = G + x$.*

Since Q is a strong module of G and $Q' = Q \cup \{x\}$ is a strong module of $G' = G + x$, then $G'/\{Q'\} = G/\{Q\}$. That is the changes implied by the insertion of x are located in T_q. Moreover, the permutation graphs family is hereditary and closed under substitution, it follows that:

Lemma 4 *$G' = G + x$ is a permutation graph iff $G'[Q'] = G[Q] + x$ is a permutation graph.*

From Lemma 4 and the discussion above, we conclude that inserting x in G reduces to insert x in $G[Q]$.

4.1.2 Modular decomposition tree of $G'[Q']$.

As the insertion node q is non-proper, it can either be: 1) a degenerate node with no mixed child but with uniform children of both types (i.e. *linked* and *notlinked*); or 2) a degenerate node with at least one mixed child; or 3) a prime node with no mixed child but a child being a twin of x in the quotient of q; or 4) a prime node with no child being a twin of x in the quotient of q. In cases 1) and 3), q is said to be *cut* (and *uncut* in cases 2) and 4)).

The case where the insertion node is a cut degenerate node (case 1) above) is similar to the case, considered by [2], of maintaining the modular decomposition tree of a cograph under vertex insertion. If q is a series (resp. parallel) node, the root q' of $T'_{q'}$ is a series (resp. parallel) node. The children of q' are those children of q typed *linked* (resp. *notlinked*) and a new parallel node q'_1. The children of q'_1 are $\{x\}$ and the remaining children of q, i.e. those typed *notlinked* (resp. *linked*).

The case where the insertion node is a cut prime node (case 3) above) is quite easy to deal with. In the children of q, the twin f of x is replaced by a new degenerate node q_1 (i.e. q_1 takes the place of f in the realiser of q). The label of q_1 is series if f is typed linked, and parallel if f is typed *notlinked*. x and f are made children of q_1.

Let us now consider the case where the insertion node q is uncut. Let us define the vertex set Q_s as the set Q if q is a prime node and as the set $F_m(q) \cup F_{nl}(q)$ (resp. $F_m(q) \cup F_l(q)$) if q a series node (resp. parallel node). The modular decomposition tree $T'_{q'}$ of $G'[Q']$ is organised as follows. If q is a prime node, then q' represents the nodes of $Q'_s = Q_s \cup \{x\}$. If q is degenerate, then q' is degenerate and has the same label than q. If q is a series (resp. parallel) node, then the set of children of q' is $\{q'_s\} \cup \mathcal{C}_l(q)$ (resp. $\{q'_s\} \cup \mathcal{C}_{nl}(q)$) where q'_s is a new node representing vertices of Q'_s. Theorem 2 states on the modular decomposition of $G'[Q'_s]$.

Theorem 2 *Let x be a vertex to be inserted in a graph G. If the insertion node q of the modular decomposition tree T of G is uncut, then $G'[Q'_s]$ is connected and co-connected. And the maximal strong modules of $G'[Q'_s]$ are $\{x\}$ and the maximal uniform (w.r.t. x) modules of $G[Q_s]$.*

Notice that the modular decomposition tree of $G'[M]$, where M is a maximal uniform module of $G[Q_s]$, is the part of T restricted to M. Therefore the whole modular decomposition tree T' of G' follows from discussion above.

4.2 Dynamic characterisation of permutation graphs

As we ask G' to be a permutation graph, the mixed nodes of T_q cannot be spread anywhere in the tree. Lemma 5 claims that there are at most two

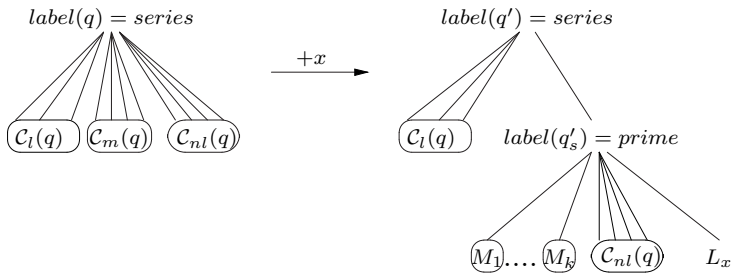

Figure 1: Updating the modular decomposition tree when the insertion node is a series node. The modules $M_1 \dots M_k$ are the maximal uniform modules of $G[Q_s]$

branches of mixed nodes in T_q beginning at q. These two branches correspond to the two families of Lemma 1.

Lemma 5 *If G' is a permutation graph then the insertion node q has at most two mixed children and any node $p \neq q$ of T_q has at most one mixed child.*

Unfortunately, Lemma 5 is not a sufficient condition for G' being a permutation graph. Theorem 3 gives necessary and sufficient conditions. Given a graph $G = (V, E)$, $S \subsetneq V$ and $y \in V \setminus S$, we denote $G - yS = (V\ ,\ E \setminus \{\{y, z\} \mid z \in S\})$. If p is a node of T_q, then set $P' = P \cup \{x\}$. Since the maximal strong modules of $G[P]$ are uniform wrt. x in $G'[P'] - xF_m(p)$, they are modules of $G'[P'] - xF_m(p)$. We denote

$$\widetilde{G'}_p = (G'[P'] - xF_m(p))/(\mathcal{MSM}(G[P]) \cup \{\{x\}\}).$$

Theorem 3 *Let x be a vertex to be inserted in a permutation graph G. Then $G' = G + x$ is a permutation graph iff either the insertion node q of the modular decomposition tree T of G is cut; or if q is uncut then:*

1. *q satisfies one of the following conditions :*

   (a) *q has two mixed children f_1 and f_2, and $\widetilde{G'}_q$ is a permutation graph admitting a realiser $\mathcal{R} = (\pi_1, \pi_2)$ such that x and f_1 are consecutive in π_1, and x and f_2 are consecutive in π_2.*

   (b) *q has a unique mixed child f_1, and $\widetilde{G'}_q$ is a permutation graph admitting a realiser $\mathcal{R} = (\pi_1, \pi_2)$ such that x and f_1 are consecutive in π_1.*

   (c) *q has no mixed child and $\widetilde{G'}_q = G'[P']/(\mathcal{MSM}(G[P]) \cup \{\{x\}\})$ is a permutation graph.*

2. *and any node* $p \neq q$ *of* T_q *satisfies one of the two following conditions :*

   (a) *p has a unique mixed child* f_1*, and* $\widetilde{G'}_p$ *is a permutation graph admitting a realiser* $\mathcal{R} = (\pi_1, \pi_2)$ *such that* x *and* f_1 *are consecutive in* π_1*, and* x *is the first element of* π_2*.*

   (b) *p has no mixed child, and* $\widetilde{G'}_p$ *is a permutation graph admitting a realiser* $\mathcal{R} = (\pi_1, \pi_2)$ *such that* x *is the first element of* π_2*.*

Due to space limitation, we only prove that the above conditions are sufficient.

Proof: $\Leftarrow$: We first show by induction that any node p of T_q different from q is such that $G'[P']$ is a permutation graph admitting a realiser $\mathcal{R}$ such that x is the first element of π_2. If p is a leaf of T_q, it trivially satisfies the inductive hypothesis. Let $p \neq q$ be a node of T_q such that its children satisfy the inductive hypothesis. If p has a unique mixed child f_1, it satisfies condition 2a of Theorem 3. According to the inductive hypothesis, $G'[F'_1]$ is a permutation graph and admits a realiser $\mathcal{R}_1 = (\tau_1, \tau_2)$ such that x is the first element of τ_2. To obtain a realiser of $G'[P']/(\mathcal{MSM}(G[P]) \setminus \{F_1\})$ such that x is the first element of π_2, the realiser $\mathcal{R} = (\pi_1, \pi_2)$ of $\widetilde{G'}_p$ is modified as follows: in π_1, substitute τ_1 for the interval $\{x, f_1\}$; and in π_2 substitute, τ_2 restricted to F_1 for f_1. Composing the resulting realiser with the realisers of the $(G[F])_{f \in \mathcal{C}(p) \setminus \{f_1\}}$, we obtain a realiser of $G'[P']$ which satisfies the inductive hypothesis. The case where p has no mixed child follows as a particular case of the previous one. This ends the induction.

If q has two mixed children f_1 and f_2, it satisfies condition 1a of Theorem 3. By the previous induction $G'[F'_1]$ and $G'[F'_2]$ are permutation graphs. They respectively admit a realiser $\mathcal{R}_1 = (\tau_1, \tau_2)$ and $\mathcal{R}_2 = (\sigma_1, \sigma_2)$ such that x is the first element of τ_2 and σ_2. In the realiser $\mathcal{R} = (\pi_1, \pi_2)$ of $\widetilde{G'}_q$, if f_2 occurs after f_1 in π_2, we reverse both orders of $\mathcal{R}_1$, and if f_2 occurs before f_1 in π_1, we reverse both orders of $\mathcal{R}_2$. To obtain a realiser of $G'[Q']/(\mathcal{MSM}(G[Q]) \setminus \{F_1, F_2\})$, $\mathcal{R}$ is modified as follows: in π_1, substitute τ_1 for the interval $\{x, f_1\}$, and σ_2 restricted to F_2 for f_2; and in π_2, substitute σ_1 for the interval $\{x, f_2\}$, and τ_2 restricted to F_1 for f_1. Composing the resulting realiser with the realisers of the $(G[V(f)])_{f \in \mathcal{C}(p) \setminus f_1, f_2}$, we obtain a realiser of $G'[Q']$. We therefore prove that $G'[Q']$ is a permutation graph. By Lemma 4 we can conclude that G is a permutation graph. The cases where p has a single or no mixed child follow as a particular cases of the above discussion. □

4.3 Algorithm and complexity

4.3.1 Data-structure.

The realiser $\mathcal{R} = (\pi_1, \pi_2)$ associated with a prime node p of the modular decomposition tree will be stored in two doubly linked lists representing the two linear orders π_1 and π_2. Each cell of a list represents a child c of p. There are two symmetric pointers between c and the cell. Moreover each cell contains its rank in the list (namely $\pi_1(c)$ or $\pi_2(c)$).

4.3.2 Routine $InsPrime$.

As a prime permutation graph G has a unique realiser $\mathcal{R} = (\pi_1, \pi_2)$, $G + x$ is a permutation graph iff x can be inserted in $\mathcal{R}$. Routine $InsPrime$ performs, if possible, that insertion.

Lemma 6 *Let $\mathcal{R} = (\pi_1, \pi_2)$ be the realiser of a prime permutation graph G and $x \notin V$ a vertex to be inserted. $G + x$ is a permutation graph iff $N(x)$ and $\overline{N}(x)$ can be respectively partitioned into $N_1(x), N_2(x)$ and $\overline{N}_1(x), \overline{N}_2(x)$ such that:*

$$\forall u_1 \in N_1(x) \cup \overline{N}_1(x), v_1 \in N_2(x) \cup \overline{N}_2(x), u_1 <_{\pi_1} v_1$$

$$\forall u_2 \in N_2(x) \cup \overline{N}_1(x), v_2 \in N_1(x) \cup \overline{N}_2(x), u_2 <_{\pi_2} v_2$$

An *initial common interval* of a realiser $\mathcal{R} = (\pi_1, \pi_2)$ is a common interval of $\mathcal{R}$ containing both $\pi_1^{-1}(1)$ and $\pi_2^{-1}(1)$. In order to find the partitions of $N(x)$ and $\overline{N}(x)$ satisfying Lemma 6, Routine $InsPrime$ makes use of the next corollary.

Corollary 1 *If $\overline{N}_1(x) \neq \varnothing$ (resp. $N_1(x) \neq \varnothing$) then $\overline{N}_1(x)$ is an initial common interval of $\mathcal{R}[\overline{N}(x)]$ (resp. $\overline{\mathcal{R}}[N(x)]$), the restriction of $\mathcal{R}$ to $\overline{N}(x)$ (resp. $N(x)$).*

Notice that an initial common interval of $\mathcal{R}[\overline{N}(x)]$ defines a partition $\overline{N}_1(x), \overline{N}_2(x)$ of $\overline{N}(x)$ (and similarly for $N(x)$). The number of initial common intervals of a realiser is $O(n)$.

Routine $InsPrime$ computes in $O(n)$ time the sets of initial common intervals of $\mathcal{R}[\overline{N}(x)]$ and of $\overline{\mathcal{R}}[N(x)]$. Then, it checks if there exists a pair of partitions $N_1(x), N_2(x)$ and $\overline{N}_1(x), \overline{N}_2(x)$ satisfying Lemma 6. Testing a given pair of partitions can be done in $O(1)$ time by comparing the ranks of the last elements of N_1 (resp. N_2) and $\overline{N}_1$ in π_1 (resp. π_2) with ranks of

the first elements of N_2 (resp. N_1) and $\overline{N}_2$. Scanning π_1, a pair of partitions satisfying the condition of Lemma 6 can be found in $O(n)$ time.

Notice that $G' = G + x$ may not be prime. If it is the case, then x has a twin vertex in G' (i.e. a vertex y s.t. $N(y) \setminus \{x\} = N(x) \setminus \{y\}$). As $\{x, y\}$ is therefore a strong module of G', by Proposition 1, x and y are consecutive in both linear orders of the realiser of G'. It follows that testing the existence of a twin can be done in $O(1)$ time if x has been inserted.

To summarise, if $G+x$ is a permutation graph, then in $O(n)$ time, Routine $InsPrime$ returns a pair of doubly linked lists, the realiser of $G + x$, and outputs the twin of x if it exists. Notice that the ranks of the cells are not maintained in these lists.

4.3.3 The typing routine.

In a bottom-up process, each node p of T receives a type (*linked*, *notlinked* or *mixed*). A leaf L_y of T is typed *linked* if $y \in N(x)$ and *notlinked* otherwise. The type of an inner node p of T depends on the types of its children. If the children of p all have the same type, p inherits that type, otherwise p is typed *mixed*. Since the number of nodes in T is $O(n)$, the typing routine runs in $O(n)$ time.

4.3.4 Finding the insertion node q.

The purpose of this step is to find the insertion node q, in the case where the root r of T is typed *mixed*. By Lemma 2, q is the *lca* of the non-proper nodes of T. Any node p of the unique path between r and q is *mixed* and proper if $p \neq q$. Since, by Definition 1, any proper mixed node has a unique mixed child, finding the insertion node can be done by a top-down search of the modular decomposition tree T. The search stops when the current node p is non-proper, which can be tested as follows. If p is a series node (resp. parallel node), then p is proper iff all its children but one are typed *linked* (resp. *notlinked*) and the remaining child is *mixed*. If p is a prime node, p is proper iff x has a twin in the quotient of p, which can be checked by Routine $InsPrime$. In both cases, testing whether p is a proper node can be done in $O(|\mathcal{C}(p)|)$. As T contains $O(n)$ nodes, the search finds the insertion node q in $O(n)$ time.

4.3.5 Maintaining the full modular representation.

We now determine if $G'[Q']$ is a permutation graph or not, and in the positive, update its full modular representation (i.e. its modular decomposition tree and the realisers of the prime nodes).

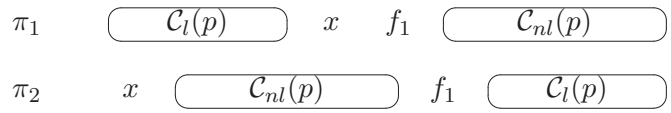

Figure 2: The unique realiser of $\widetilde{G}'_p$ (if p is a series node) that fulfils condition 2a of Theorem 3. For a parallel node p, $\mathcal{C}_{nl}(p)$ and $\mathcal{C}_l(p)$ has to be exchanged in π_2.

If the insertion node q has more than two mixed children, from Lemma 5, $G'[Q']$ is not a permutation graph: the algorithm stops. If q is cut, from earlier discussion $G'[Q']$ is always a permutation graph (see Section 4.1). In that case, the realisers of the prime nodes are not modified. Therefore $T'_{q'}$ can be computed in $O(|\mathcal{C}(q)|)$ as described in Section 4.1. When q is uncut, the nodes of T_q have to fulfil the conditions stated in Theorem 3. To simplify the presentation, let us present our algorithm as three-step process. But notice in practice these three steps can be merged into a single one.

- For each node p of T_q, we check whether p fulfils the condition of Theorem 3. If p is a degenerate node having the right number of mixed children (0, 1 or 2 depending on $p = q$), then $\widetilde{G}'_p$ always enjoys a realiser satisfying Theorem 3 (see Figure 2). If p is prime, using Routine $InsPrime$, we insert x in the realiser associated to p by making x adjacent to $\mathcal{C}_l(p)$ and non-adjacent to $\mathcal{C}_m(p) \cup \mathcal{C}_{nl}(p)$. There may be two different positions to insert x (only if has a twin vertex). We then test if at least one of the possible positions fulfils the conditions of Theorem 3 which simply consists in testing the position of x in the realiser returned by $InsPrime$ (extremity in an order and/or consecutiveness with the mixed children). That can be done in $O(1)$. Since we handle only the quotients of the nodes p of T_q, each of which being processed in $O(|\mathcal{C}(p)|)$ time, this first steps runs in $O(n)$ time.

- Theorem 2 states that the maximal strong modules of $G'[Q'_s]$ are $\{x\}$ and the maximal uniform modules of $G[Q_s]$. These maximal uniform modules can be found in $O(n)$ time by a search in T_q since M is a maximal uniform module iff there exists a mixed node p descendant of the insertion node q such that either p is degenerate and $M = F_l(p)$ or $M = F_{nl}(p)$; or p is prime and M is the vertex set of some uniform child of p. By Theorem 2, these modules will be represented by the children nodes of the new prime node q'_s. Recall that the modular decomposition tree of $G'[M]$ is inherited from the modular decomposition tree T of G.

- The last step computes the realiser $\mathcal{R}_s$ of the quotient of $G'[Q'_s]$ by its maximal strong modules. Notice that in the intermediate realisers computed along the process, the ranks of the cells in the lists are not maintained.

  To that aim, we applied the bottom-up process, described in the proof of Theorem 3, on the modular decomposition tree T where each maximal uniform module has first been contracted into a single vertex (i.e. replaced by a leaf in the tree T). For a prime mixed node p, the realiser of $\widetilde{G}'_p$ is given by Routine $InsPrime$. For a degenerate node p, the realiser of $\widetilde{G}'_p$ is the one depicted in Figure 2. As the realisers are encoded by pairs of doubly linked lists, the substitution operation used in the proof of Theorem 3 can be done in $O(1)$ time. It follows that the realiser $\mathcal{R}_s$ can be computed in $O(n)$ time.

  Finally to maintain the data-structure, a scan of the lists of $\mathcal{R}_s$ allows to get the ranks of the cells.

Theorem 4 *Updating the full modular representation of a permutation graph under vertex insertion costs $O(n)$ time.*

References

[1] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: a Survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.

[2] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear time recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[3] C. Crespelle and C. Paul. Fully-dynamic recognition algorithm and certificate for directed cographs. In *30th Int. Workshop on Graph Theoretical Concepts in Computer Science (WG04)*, number 3353 in Lecture Notes in Computer Science, pages 93–104, 2004.

[4] F. de Montgolfier. *Décomposition modulaire des graphes - Théorie, extensions et algorithmes*. PhD thesis, Université de Montpellier 2, France, 2003.

[5] Tibor Gallai. Transitiv orientierbare graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.

[6] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[7] P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM Journal on Computing*, 31(1):289–305, 2002.

[8] L. Ibarra. Fully dynamic algorithms for chordal graphs. In *10th ACM-SIAM Annual Symposium on Discrete Algorithm (SODA'03)*, pages 923–924, 1999.

[9] J.H. Muller and J.P. Spinrad. Incremental modular decomposition algorithm. *Journal of the Association for Computing Machinery*, 36(1):1–19, 1989.

[10] R. Shamir and R. Sharan. A fully dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Applied Mathematics*, 136(2-3):329–340, 2004.

[11] Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.