Fully-Dynamic Recognition Algorithm and Certificate for Directed Cographs

Christophe Crespelle, Christophe Paul

Fully-dynamic recognition algorithm and certificate for directed cographs

Christophe Crespelle and Christophe Paul

CNRS - LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France,
{crespell,paul}@lirmm.fr

Abstract. This paper presents an optimal fully-dynamic recognition algorithm for directed cographs. Given the modular decomposition tree of a directed cograph G, the algorithm supports arc and vertex modification (insertion or deletion) in $\mathcal{O}(d)$ time where d is the number of arcs involved in the operation. Moreover, if the modified graph remains a directed cograph, the modular tree decomposition is updated; otherwise, a certificate is returned within the same complexity.

1 Introduction

Directed cographs is the family of digraphs recursively defined from the single vertex under the closure of the operations of *disjoint union*, *series* and *order*. Let $G_1, \ldots, G_k$ be a set of k disjoint digraphs. The *disjoint union* (or *parallel* composition) of the G_i's is the digraph whose connected components are precisely the G_i's. The *series* composition of the G_i's is the union of these k graphs plus all possible arcs between vertices of different G_i's. The *order* composition of the G_i's is the union of these k graphs plus all possible arcs from G_i towards G_j, with $1 \leqslant i < j \leqslant k$. These operations define a unique tree representation of the directed cograph referred which corresponds to its modular decomposition tree [12]. The leaves are mapped to the vertices of the graph and the inner nodes are labeled by the different composition operations (see Figure 1). Notice that by definition of the composition operations, the complement of a directed cograph is a directed cograph. Indeed, the term *cograph* [3] stands for *complement reducible graph*. Moreover the directed cograph family is *hereditary*: any induced subgraph of a directed cograph is also a directed cograph. It should also be noticed that directed cographs can be characterized by forbidden subgraphs (see Theorem 2 and Figure 2).

Restricted to posets, directed cographs are the *series-parallel orders* [11] for which the recognition problem has been solved in linear time [14]. In the case of undirected graphs, the series composition and the order composition are equivalent. The family of undirected graphs defined from the single vertex graph by the closure of the series and the disjoint composition is the *cographs*. The modular decomposition tree of a cograph is called a *cotree*. A number of linear time cograph recognition algorithms is now known: the first one was presented in [4] and the most recent one in [1].

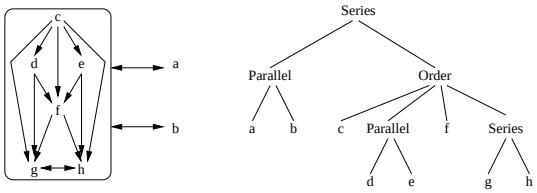

Fig. 1. A directed cograph and its modular decomposition tree. Since set $\{a, b\}$ is in series composition with the rest of the vertices, for any $x \notin \{a, b\}$ and $y \in \{a, b\}$, both arcs xy and yx exist.

The *dynamic recognition and representation problem* 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: adding a vertex (along with the arcs incident to it), deleting a vertex (and its incident arcs), adding or deleting an arc or two symmetric arcs (notice that the insertion/deletion of only one of these arcs may not result in a graph of $\mathcal{F}$, while the insertion/deletion of both would). Moreover, as pointed out by [10], if the property of belonging to $\mathcal{F}$ is no longer satisfied, providing a certificate would be highly desirable in practice (eg. for debugging features). This paper considers that problem for the family of directed cographs. The representation we maintain is the modular decomposition tree.

Related works Several authors have considered the dynamic recognition and representation problem for various graphs families. [9] devised a fully dynamic recognition algorithm for chordal graphs which handles edge operations in $\mathcal{O}(n)$ time. For proper interval graphs [8], each update can be supported in $\mathcal{O}(d + \log n)$ where d is the number of edges involved in the operation. Concerning cographs, a constant time algorithm for edge modification (insertion or deletion) has been designed in [13]. The undirected cograph recognition algorithm of [4] is incremental: given a cograph G, its cotree T and a vertex x, it modifies T iff $G + x$ is a cograph. Merging the results of [4] and [13] provides a fully dynamic recognition algorithm for cographs with $\mathcal{O}(d)$ worst case time complexity per operation. Pushing further Algorithm of [4], if $G + x$ is not a cograph, it is possible, within the same complexity, to extract a certificate (namely a P_4, an induced path of 4 vertices).

The work of [4] has recently been extended for bipartite graphs. A new decomposition dedicated to bipartite graphs has been proposed in [6] and the family of bipartite graphs totally decomposable, as are the cographs for the modular decomposition, are defined: the weak-bisplit graphs. In [7], a linear time recognition algorithm of weak-bisplit graphs is given. It turns out that the incidence bipartite graph of a directed cograph is a weak-bisplit graph. As for cographs, the decomposition tree is built by adding the vertices one by one. But unfortu-

nately, to get linear time complexity, the vertices have to be ordered with respect to their degree. It follows that the incremental aspect cannot be guaranteed.

Our results We present an optimal algorithm for the dynamic recognition and representation problem for the family of directed cographs. If needed, our algorithm is also able to find a certificate. Therefore, it extends the algorithms of [4, 13]. Moreover, unlike the algorithm of [7] restricted to directed cographs, our algorithm supports arc modification and the dynamical aspect is guaranteed (that is the updates can be handled in arbitrary order).

Theorem 1. *The dynamic recognition and representation problem for directed cographs is solvable in $\mathcal{O}(d)$ worst case time per update, where d is the number of edges involved in the updating operation. Moreover, if needed, a certificate that the modified graph is not a directed cograph, is provided within the same time complexity.*

Note that the results of [4] and [13] for undirected cographs cannot solve the directed case since there is no way, to our knowledge, to determine if an orientation of an undirected cograph is a directed cograph or not.

2 Preliminaries

Any graph $G = (V, E)$ considered here will be finite, loopless and directed, with $n = |V|$ and $m = |E|$. The complement of a graph G is denoted by $\overline{G}$. If X is a subset of vertices, then $G[X]$ is the subgraph of G induced by X. Since the graphs are directed, the arc xy differs from yx. Let x be a vertex, then $N^+(x) = \{z \in V, xz \in E\}$, $N^-(x) = \{y \in V, yx \in E\}$ and $N(x) = N^-(x) \cup N^+(x)$ stand respectively for the *out-neighborhood* of x, its *in-neighborhood* and its neighborhood. The non-neighborhood of x will be designed by $\overline{N}(x)$. The degree $d(x)$ of a vertex x is the sum of its in-degree, $d^-(x) = |N^-(x)|$, and its out-degree, $d^+(x) = |N^+(x)|$. Let $G = (V, E)$ be a digraph, $x \notin V$ be a vertex and $N^-(x) \subseteq V$, $N^+(x) \subseteq V$ be its in and out-neighorhoods. Then $G + x$ denotes the digraph $G' = (V \cup \{x\}, E \cup \{xz, z \in N^+(x)\} \cup \{yx, y \in N^-(x)\})$. If $xy \in E$, $G - xy$ will be the digraph $G' = (V, E \setminus \{xy\})$. $G - x$ and $G + xy$ are similarly defined.

As for the cographs family, directed cographs can be characterized by forbidden subgraphs. Unfortunately, such a characterization does not help for an efficient recognition algorithm (even for a non-dynamical one). Nevertheless, these subgraphs will be useful to provide a certificate if the referred graph is not a directed cograph. This characterization can be retrieved from a result of [5].

Theorem 2. *A digraph is a directed cograph iff it does not contain any graph of Figure 2 as induced subgraph.*

A *module* M is a set of vertices such that for any $x \notin M$ and $y \in M$, $xy \in E$ iff $\forall z \in M$, $xz \in E$ and $yx \in E$ iff $\forall z \in M$, $zx \in E$. The modules

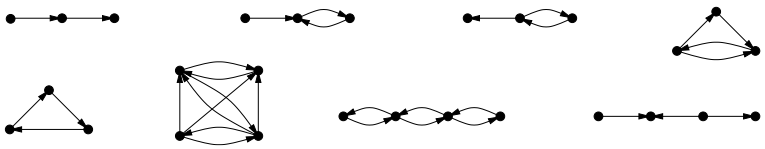

Fig. 2. The set of forbidden subgraphs for the directed cographs family. Notice that this set is closed under complementation.

of a graph are a potentially exponential-sized family. However, the sub-family of *strong* modules, the modules that overlap[1] no other module, has size $O(n)$. The inclusion order of this family defines the *modular decomposition tree*, which is enough to store the module family of a graph [12]. The root of this tree is the trivial module V and its n leaves are the trivial modules $\{x\}, x \in V$. In the case of directed cographs, the internal nodes are labeled by one of the three composition operations: *parallel* (*disjoint union*), *series* or *order* (see Figure 1). Let us call the modular decomposition tree of a directed cograph, the *di-cotree*.

Any node p of the di-cotree corresponds to a set of vertices $M(p)$. To shorten the notations, the set $M(p)$ will be denoted by P. A set $S \subseteq V$ of vertices is *uniform* wrt. $x \notin S$ in G if S is a module of the graph $G[S \cup \{x\}]$. If S is not uniform, then it is *mixed*. We say that p is uniform (resp. mixed) wrt. x if P is. Finally, a set S of vertices (resp. a node p of the di-cotree) is *linked* to a vertex $x \notin S$ in G, if there exists $y \in S$ (resp. $y \in P$) st. $xy \in E$ or $yx \in E$. In the following, if no confusion is possible, we will omit to mention the graph in which the above notions are applied. The subtree of the di-cotree T rooted at a node q will be denoted by T_q. The path between any node p and the root r of T will be denoted by P_q^r. Finally, M_{xy} stands for the minimum (wrt. the inclusion order) module that contains vertices x and y. Since M_{xy} is not necessarily strong, it is a subset of $M(p_{xy})$ where p_{xy} is the least common ancestor of the leaves corresponding to x and y. A *factorizing permutation* [2] τ is a permutation of the vertices such that any strong module M is a factor of τ (the vertices of M occur consecutively). A DFS of the modular decomposition tree orders the leaves as a factorizing permutation. Maintaining factorizing permutation is helpful to find a certificate.

3 Dynamic vertex operations

This section deals with Theorem 1 in the case of vertex modification (insertion or deletion). Vertex deletion is first considered. Then a theorem characterizes the cases where the insertion of a vertex is possible. This theorem is the basis of an insertion algorithm that either updates the di-cotree or finds a certificate that G is not a directed cograph. For sake of simplicity, the certificate consists in a set of 4 vertices that induces a subgraph containing a forbidden subgraph. Pushing further the algorithm, an exact forbidden subgraph can be found.

[1] A overlaps B if $A \cap B \neq \emptyset$, $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$

3.1 Deleting a vertex

As already noticed, the family of directed cographs is hereditary. It follows that deleting a vertex of a directed cograph G only requires to update its di-cotree T. It can be done in $\mathcal{O}(d(x))$ as follows (see [13] for a similar algorithm). The case where x is the only vertex is trivial. Otherwise, let q be the parent node of x in T.

1. If x has at least 2 siblings, then x is removed from T.
2. Otherwise, let p be the sibling of x.
   (a) If q is the root of T or the label of $parent(q) = \tilde{q}$ is different from the one of p, x is removed from T and q replaced by p.
   (b) If $label(\tilde{q}) = label(p)$, nodes x, q and p are removed from T and q is replaced by the children of p, respecting their relative order if p is an order node.

For complexity issues, the case where $p, \tilde{q}$ have the same label, has to be handled carefully: only nodes containing neighbors of x can be touched. If q is a parallel node, its siblings are disconnected from $\tilde{q}$ and reconnected as new children of p (at their right place if $\tilde{q}$ is an order node). Finally p replaces $\tilde{q}$. If q is not a parallel node, the children of p can be moved similarly.

3.2 Adding a vertex

The main difficulty consists in maintaining a di-cotree under vertex insertion. Theorem 3 characterizes the cases where given a directed cograph G, a vertex x and its neighborhoods, the augmented graph $G + x$ remains a directed cograph. As in [4], the algorithm first proceeds a marking step of the di-cotree T of G. Then it tests whether the marks satisfy Theorem 3. In the positive, the di-cotree is updated; otherwise a certificate that $G + x$ is not a directed cograph is given.

Theorem 3. *Let $G = (V, E)$ be a directed cograph and T be its di-cotree. Let $x \notin V$ be a vertex and $N^-(x)$, $N^+(x)$ be its in and out-neighborhoods. $G' = G+x$ is a directed cograph iff for any node p of T one of the following conditions holds:*

1. *P is uniform wrt. x;*
2. *P is mixed and has a unique mixed child f such that $F \cup \{x\}$ is a module of $G'[P \cup \{x\}]$;*
3. *P is mixed, has no mixed child and either*
   (a) *there exists a unique non-empty set $\mathcal{S} \subset \mathcal{C}(p)$ of children of p such that $S = \bigcup_{k \in \mathcal{S}} K$ is uniform wrt. x and $S \cup \{x\}$ is a module of $G'[P \cup \{x\}]$,*
   (b) *or there exists a non-empty set $\mathcal{S} \subset \mathcal{C}(p)$ of children of p such that $S \cup \{x\}$, $(P \setminus S) \cup \{x\}$ are both modules of $G'[P \cup \{x\}]$.*

Corollary 1 shows that the mixed nodes cannot be spread anywhere in T.

Corollary 1. *Assume $G + x$ is a directed cograph. The set of mixed nodes induces a path between the root and a certain node p of the di-cotree. Node p is the only mixed node without mixed child.*

Before describing the marking process, let us rephrase Theorem 3. Hereafter the only mixed node without mixed child will be called the *terminal mixed node* of T. A *single mixed node* will be a node satisfying condition `2` of Theorem 3. As illustrated by Figure 3, it is worth to notice that case `3.a` and `3.b` of Theorem 3 are exclusive. Indeed, in case `3.a`, x is not a maximal (wrt. inclusion order) strong module of $G[P \cup \{x\}]$: therefore x will be inserted as a grand-child of the terminal mixed node p. While in case `3.b`, x is a maximal strong module of $G[P \cup \{x\}]$ and should be inserted as a child of p. Moreover in the last case, p is an order node.

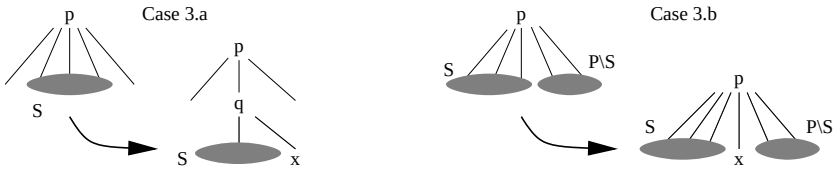

Fig. 3. Modifications of the modular decomposition tree according to cases `3` of Theorem 3. In case `3.b`, the node p is an order node. In case `3.a`, depending on the cardinality of $\mathcal{S}$, the label of the nodes of $\mathcal{S}$ and their adjacency with x, intermediate nodes may be inserted between q and S (see Subsection *Inserting a vertex*).

The marking process The first step of our algorithm colors nodes of the modular decomposition tree T according to the neighborhood of the vertex to be inserted. This preliminary step can be seen as an extension of the marking process of [4].

Initially each leaf $l = \{y\}$, such that $y \in N(x)$, is colored red. Depending on the adjacency relationship between y and x, these leaves are given a type: $type(l) = InOut$ if $xy \in E$ and $yx \in E$; $type(l) = In$ if $yx \in E$; or $type(l) = Out$ if $xy \in E$. The process is a bottom-up search: each red node p forwards its type to its parent node q and depending on the different types received by q, a color is given to q. The first time a node receives a forwarded type from one of its children, it is colored black. A node q becomes red if all its children are of the same type (ie. the corresponding set of vertices Q is uniformly linked to x). A red node receives the type of its children. Once a red node has been processed it becomes grey. In order to prepare the possible insertion of x, a list of the grey children is maintained for each node.

For sake of simplicity, let us say that the default color is *white*. Also notice that the absence of type can be considered as a non-adjacency type, we will use the notation $type(p) = None$. However, the marking algorithm will never manage neither the white nodes nor the $None$ type.

Each node stores the list of its grey children and a few counters: eg. $\#child(q)$ indicates the number of children of q, $\#type(q, In)$ the number of children of q whose type is In. It is straightforward to see that the running time of Routine `Type` is $\mathcal{O}(d(x))$. Let T^c be the resulting colored di-cotree. The number of grey nodes and of black nodes are both bounded by $\mathcal{O}(d(x))$.

```
Type(G, T, R a set of typed red leaves)
 1. While some red node p exists Do
 2.       color(p) ← grey
 3.       If p is not the root of T Then
 4.             Let q be the parent node of p
 5.             Add p to the list greyChild(q)
 6.             Increase #type(q, type(p)) by one
 7.             If #type(q, type(p)) = #child(q) Then
 8.                   color(q) ← red and type(q) ← type(p)
 9.             Else color(q) ← black
10. End of while
```

Fig. 4. Marking process.

Lemma 1. *If there exists a black node q such that:*

1. *any black node belongs to P_q^r,*
2. *any black node of $P^r_{parent(q)}$ is series or order and*
3. *any white node of $P^r_{parent(q)}$ is parallel,*

then the white nodes of P_q^r are single mixed nodes.

The set of black nodes will be denoted $\mathcal{B}$. By definition, a black node is a mixed node. A white node p can be mixed if T_p contains a black node, otherwise it is uniform. Lemma 1 implies that the set of white mixed nodes is exactly the set $\mathcal{W}$ of white nodes of the path P_q^r mentioned in Lemma 1. Therefore $\mathcal{W} \cup \mathcal{B}$ is the set of mixed nodes of T^c.

Testing the insertion Assume the following conditions are satisfied: there exists a terminal mixed node $q \in \mathcal{B}$ such that any node of $\mathcal{B}$ belongs to P_q^r and any nodes of $P^r_{parent(q)}$ is a single mixed node. Then by Lemma 1, any node of T satisfies the hypothesis of Theorem 3. It implies that x can be inserted. Therefore Routine `Check` (see Figure 5) only has to test these conditions. If one of them is not satisfied, then a call to Routine `Find-Certificate` enables us to find a set Z of 3 vertices such that $G'[Z \cup \{x\}]$, with $G' = G + x$, contains one of the forbidden subgraphs of Figure 2. The insertion of vertex x, if possible, is handled by Routine `Insert`.

Let p be the current node in Routine `Check`. If p has already been visited (test Line 6), then by Corollary 1 G' is not a directed cograph. The tests of Line 7 and 8 check whether p is a single mixed node. As shown by Lemma 2, depending on their color, the label of single mixed nodes are constrained.

Lemma 2. *Let p be a single mixed node of T^c. If p is black, then p is either a series or an order node. Otherwise it is a white parallel node.*

Check(G, T^c, $\mathcal{B}$, x)
1. $bottom \leftarrow r$, where r is the root of T
2. **While** some node q in $\mathcal{B}$ exists **Do**
3. $p \leftarrow q$ and remove q from $\mathcal{B}$
4. **While** $p \neq bottom$ **Do**
5. $p \leftarrow parent(p)$
6. **If** p has been visited **Then** `Find-Certificate`(p)
7. **If** p is a white non-parallel node **Then** `Find-Certificate`(p)
8. **If** $p \in \mathcal{B}$ is not a single mixed node **Then** `Find-Certificate`(p)
9. **If** $p \in \mathcal{B}$ **Then** Remove p from $\mathcal{B}$
10. Mark p as visited
11. **End of while**
12. $bottom \leftarrow q$
13. **End of while**
14. **If** q is a terminal mixed node **Then** `Insert`(x, q, T)
15. **Else** `Find-Certificate`(q)

Fig. 5. Testing the insertion.

Let p be a black (series or order) node. For p to be a single mixed node, all but one of its children have to be colored grey. If p is a series node, the children distinct from the only non-grey child q should be typed $InOut$. If p is an order node, the children that occur before (resp. after) q have to be typed In (resp. Out).

Finally let q be the last node considered by routine `Check` ($bottom$). There is no constraint on the label of q. By Theorem 3, it has to be terminal mixed, which can be tested as follows:

- if q is a parallel node: check that $\#type(In) = \#grey$ or $\#type(Out) = \#grey$ or $\#type(InOut) = \#grey$ (since in that case, any node of $\mathcal{S}$ is a grey node);
- if q is a series node: check that $\#type(In) + \#type(InOut) = |\mathcal{C}(q)|$ or $\#type(Out) + \#type(InOut) = |\mathcal{C}(q)|$ or $\#type(In) = \#type(Out) = 0$;
- if q is an order node: first, test if either $\#type(InOut) + \#type(In) + \#type(Out) = |\mathcal{C}(q)|$ or $\#type(InOut) = 0$. Then check whether the first (wrt. the relative order of q) $\#type(In)$ children of q are typed In and the last $\#type(Out)$ are typed Out.

If each node p of the di-cotree stores its number of children $|\mathcal{C}|$, these tests can be done by a simple search in the grey children. Since the number of grey nodes and black nodes is $\mathcal{O}(d(x))$, Routine `Check` runs in $\mathcal{O}(d(x))$ time.

Inserting a vertex As illustrated by Figure 3, the modification occurs in the di-cotree T_q where q is the only terminal mixed node (the $bottom$ node in Routine `Check`). We know that any child of q is uniform and since q is mixed, it has at

least two children of different types (remind that the absence of type can be considered as a non-adjacency type).

Assume q is a series node (the case where q is parallel is similar). As already noticed, since q is not an order node, x has to be inserted as a grand-child of q. By theorem 3, a set $\mathcal{S}$ of children such that $S = \bigcup_{k \in \mathcal{S}} K$ is uniform wrt. x and $S \cup \{x\}$ is a module of $G'[Q \cup \{x\}]$, where $G' = G + x$, exists. Since q is a series node, a child p belongs to $\mathcal{S}$ iff $type(p) \neq InOut$. Remark that the uniformity of S implies that the nodes belonging to $\mathcal{S}$ all have the same type. To update the di-cotree, three cases should be considered. First, if $\mathcal{S} = \{p\}$ and the label of node p coincides with its type, $type(p)$, then x is added as a leaf of p. Otherwise, a new node p', labeled by the composition operation corresponding to the type of nodes of $\mathcal{S}$, is inserted as a child of q instead of nodes of $\mathcal{S}$. If $\mathcal{S} = \{p\}$, then p is made a child of this new node p'. If $|\mathcal{S}| \geq 2$, a node q', whose children are the nodes of $\mathcal{S}$, is made a child of p. q' get the same label than q (ie. series). In both cases, x is added as a leaf of p'.

Assume q is an order node. The difference with the previous cases, is that three subsets of $\mathcal{C}(q)$ can be identified: $\mathcal{S}_{In}$ (resp. $\mathcal{S}_{Out}$) the children with type In (resp. Out) and $\mathcal{S}$ the other children. The nodes of $\mathcal{S}_{In}$ appears before those of $\mathcal{S}$ that appears before those of $\mathcal{S}_{Out}$ in the order defined by q. Notice that one of these three sets could be empty. If $\mathcal{S} = \emptyset$, x is inserted as a child of q between $\mathcal{S}_{In}$ and $\mathcal{S}_{Out}$. Otherwise, a new child p of q has to be inserted between q and $\mathcal{S}$, and x is made a child of p.

As done for the vertex deletion, to update the di-cotree, we have to carefully handle the moving of non-neighborhood of x. Any insertion costs $\mathcal{O}(d(x))$ time.

Finding a certificate Routine `Find-Certificate`(p) looks for one of the forbidden induced subgraphs of Figure 2. Assume this routine is also given the parameters P^r_{bottom} and P^p_q where *bottom* and q are the nodes respectively defined at Line 12 and 2 of Routine `Check`. Thanks to the lists of grey children for each node of T^c and the factorizing permutation, the search is processed in $\mathcal{O}(d(x))$ time. The call to `Find-Certificate` at Line 6 occurs if the current node p has already been visited before. At Lines 7 and 8, node p should have been a single mixed node but is not. At Line 15, the last visited node q is not terminal mixed. In each case, a subgraph of Figure 2 can be found in $\mathcal{O}(d(x))$ time. Though Routine `Find-Certificate` returns the exact subgraph, for sake of simplicity, we just give some hints of the following:

Lemma 3. *If $G' = G + x$ is not a directed cograph, a set Z of 3 vertices can be found in $\mathcal{O}(d(x))$ time such that $G'[Z \cup \{x\}]$ contains one of the graphs of Figure 2.*

Let us detail the former call of Line 6. Notice that p has to be a parallel node (otherwise, `Check` would have found out that p is not a single mixed node). Indeed, p has at least 2 black mixed children: namely h, the child of p on the path P^r_{bottom}, and h', the child of p on the path P^p_q. Since h and h' are black, they both received a type from a grey child, say k and k' respectively. Let a be a vertex of

$K = M(k)$ and b be a vertex of $K' = M(k')$. Finally, since h' is mixed and k' is uniform, a vertex $c \in H' \setminus K'$ (with $H' = M(h')$) such that $type(c) \neq type(b)$ exists. A simple case by case analysis of the different possible types for nodes k, k' and vertices a, b, c proves that $G'[\{a, b, c, x\}]$, with $G' = G + x$, contains a certificate (one of the graphs of Figure 2). Figure 6 illustrates two different cases.

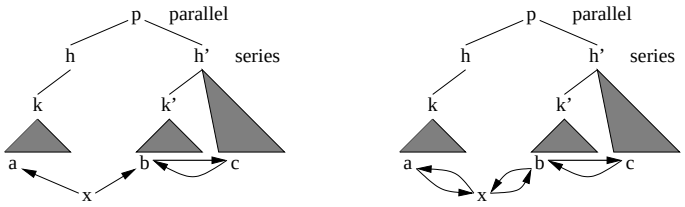

Fig. 6. Since p is a parallel node, h' is either a series or an order node. Assume that h' is a series nodes, therefore bc and cb exist. In the first example, the certificate is induced by $\{b, c, x\}$, in the second by $\{a, b, c, x\}$.

4 Dynamic arc operations

This section deals with Theorem 1 in the case of arc modification. For lack of space, we only present how to handle arc deletion. Since the family of directed cographs is closed under complementation, the graph $G+xy$ is a directed cograph iff the graph $\overline{G} - xy$ is. Similarly, a certificate that $\overline{G} - xy$ is not a directed cograph, is a certificate for $G + xy$. As remarked by [13], since the di-cotree of the complement of a directed cograph G can easily be deduced from the di-cotree of G a recognition algorithm that supports edge deletion can be extended to support edge insertion within the same complexity. Finally, we shall assume that if an arc deletion query is asked for, then the arc involved exists.

Deleting an arc Two types of arc based modifications should be distinguished. The first one concerns the simultaneous removal of two symmetric arcs, say xy and yx. This modification can be compared to the deletion of an edge in an undirected cograph, see [13]. The proof of [13] can be generalized to the case of directed cographs. Let q_x (resp. q_y) be the child of p_{xy} containing x (resp. y). [2]

Theorem 4. *[13] The graph $G' = G - \{xy, yx\}$ is a directed cograph iff $|Q_x| = 1$ and $Q_y \setminus \{y\} \subseteq \overline{N}(y)$ or $|Q_y| = 1$ and $Q_x \setminus \{x\} \subseteq \overline{N}(x)$.*

Theorem 5 extends Theorem 4 so that any valid arc modification of a directed cograph can be characterized.

Theorem 5. *The graph $G' = G - xy$ is a directed cograph iff*

[2] p_{xy} is the lca of x and y in T

1. *p_{xy} is an order node, $M_{xy} = Q_x \cup Q_y$ and:* [3]
   (a) *either $|Q_x| = 1$ and $Q_y \setminus \{y\} \subseteq \overline{N}(y)$,*
   (b) *or $|Q_y| = 1$ and $Q_x \setminus \{x\} \subseteq \overline{N}(x)$.*
2. *p_{xy} is a series node and:*
   (a) *either $|Q_x| = 1$ and $Q_y \setminus \{y\} \subseteq N^+(y) \setminus N^-(y)$,*
   (b) *or $|Q_y| = 1$ and $Q_x \setminus \{x\} \subseteq N^-(x) \setminus N^+(x)$.*

It is straightforward from Theorem 4 and 5 that the deletion test can be done in $\mathcal{O}(1)$. Indeed, x and y has to be the child and the grand-child of p_{xy}. Wlog. assume x is the child. Then, it suffices to check the label of p_{xy} and of its child q which is the parent of y. If the deletion is possible, the modifications of the di-cotree are carried out in constant time. The different cases are depicted in Figure 7.

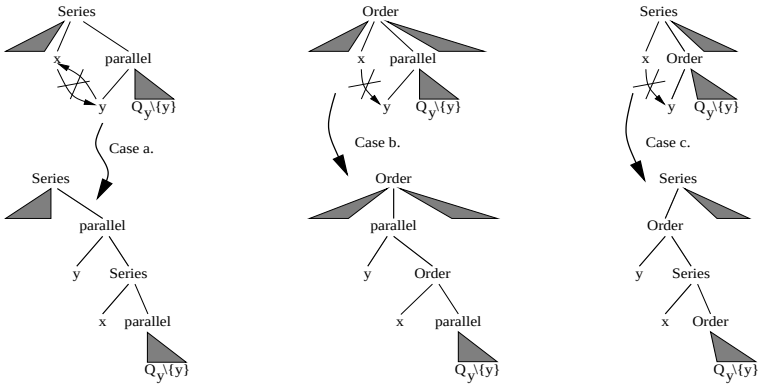

Fig. 7. Case `a.` illustrates the modification implied by the simultaneous removal of two symmetric arcs (see Theorem 4); cases `b.` and `c.` illustrate the removal of the arc xy described in Theorem 5. Depending on the number of siblings of y, the resulting di-cotrees may contain less node than above.

Finding a certificate Assume the test of the xy deletion fails. As done for the vertex certificate, our algorithm returns a small subgraph containing one of the graphs of Figure 2. Thanks to the factorizing permutation, the vertices of this subgraph can be found in constant time. If an exact certificate is wished, it can be found in $\mathcal{O}(min(d(x), d(y)))$.

Lemma 4. *If $G' = G - xy$ is not a directed cograph, a set Z of at most 4 vertices can be found in $\mathcal{O}(1)$ such that $G'[Z \cup \{x, y\}]$ contains one of the graphs of Fig. 2.*

Let us describe how the set Z is defined. Let p_x (resp. p_y) be the parent of x (resp. y) in T. If $p_x \neq r$ (resp. $p_y \neq r$), let q_x (resp. q_y) be the parent of p_x

[3] M_{xy} is defined as the minimum module containing x and y. Therefore $M_{xy} \subseteq P_{xy}$.

(resp. p_y) in T. If $q_x \neq r$ (resp. $q_y \neq r$), let k_x (resp. k_y) be the parent of q_x (resp. q_y) in T. Let us define 6 vertices, namely a_x, b_x, c_x and a_y, b_y, c_y. Vertex a_x belongs to $P_x \setminus \{x\}$ and if p_x is an order node and $P_x \cap N^+(x) \neq \emptyset$, then choose for a_x an out-neighbor of x. Vertices b_x and c_x belongs respectively to $Q_x \setminus P_x$ and $K_x \setminus Q_x$ if these sets exist. The last 3 vertices a_y, b_y, c_y are similarly defined wrt. y. If possible, a_y should be picked in $N^-(y)$. Note that, even if they exist, these vertices may not be all distinct. Finally a case by case analysis of the labels of parents and grand-parents of x and y enables us to select at most 4 vertices among $a_x, a_y, b_x, b_y, c_x, c_y$.

References

1. A. Bretscher, D.G. Corneil, M. Habib, and C. Paul. A simple linear time lexbfs cograph recognition algorithm. In H. Bodlaender, editor, *29th International Workshop on Graph Theoretical Concepts in Computer Science (WG'03)*, number 2880 in Lecture Notes in Computer Science, pages 119–130, 2003.
2. C. Capelle and M. Habib. Graph decompositions and factorizing permutations. In *Fifth Israel Symposium on the Theory of Computing Systems (ISTCS'97)*, IEEE Computer Society, pages 132–143, 1997.
3. D.G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(1):163–174, 1981.
4. 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.
5. A. Ehrenfeucht and G. Rozenberg. Primitivity is hereditary for 2-structures. *Theoretical Computer Science*, 70(3):343–359, 1990.
6. J.-L. Fouquet, V. Giakoumakis, and J.-M. Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International Journal of Foundation of Computer Science*, 10(4):513–533, 1999.
7. V. Giakoumakis and J.-M. Vanherpe. Linear time recognition and optimizations for weak-bisplit graphs, bi-cographs and bipartite p_6-free graphs. *International Journal of Foundation of Computer Science*, 14(1):107–136, 2003.
8. 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.
9. L. Ibarra. Fully dynamic algorithms for chordal graphs. In *10th ACM-SIAM Annual Symposium on Discrete Algorithm (SODA'03)*, pages 923–924, 1999.
10. D. Kratsch, R.M. McConnell, K. Mehlhorn, and J.P. Spinrad. Certifying algorithm for recognition of interval graphs and permutation graphs. In *14th ACM-SIAM Annual Symposium on Discrete Algorithm (SODA'03)*, pages 153–167, 2003.
11. E.L. Lawler. Graphical algorithm and their complexity. *Mathematical center tracts*, 81:3–32, 1976.
12. R.H. Möhring and F. J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.
13. R. Shamir and R. Sharan. A fully dynamic algorithm for modular decomposition and representation of cographs. *Discrete Applied Mathematics*, 136(2-3):329–340, 2004.
14. J. Valdes, R.E. Tarjan, and E.L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11:298–313, 1982.