REPORTS
IN
INFORMATICS

ISSN 0333-3590

Characterizing and computing minimal cograph completions

Daniel Lokshtanov Federico Mancini
Charis Papadopoulos

REPORT NO 352 February 2008

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

Characterizing and computing minimal cograph completions [*]

Daniel Lokshtanov[†] Federico Mancini[†] Charis Papadopoulos[†]

Abstract

A cograph completion of an arbitrary graph G is a cograph supergraph of G on the same vertex set. Such a completion is called minimal if the set of edges added to G is inclusion minimal. In this paper we present two results on minimal cograph completions. The first is a a characterization that allows us to check in linear time whether a given cograph completion is minimal. The second result is a vertex incremental algorithm to compute a minimal cograph completion H of an arbitrary input graph G in $O(|V(H)| + |E(H)|)$ time.

1 Introduction

Cographs are exactly the graphs with no induced path on four vertices. Any graph can be embedded into a cograph by adding edges to the original graph and the resulting graph is called a *cograph completion*, whereas the added edges are called *fill edges*. A cograph completion with the minimum number of edges is called *minimum*, while it is called *minimal* if no proper subset of the fill edges produces a cograph when added to the original graph.

Computing a minimum completion of an arbitrary graph into a specific graph class is an important and well studied problem with applications in molecular biology, numerical algebra, and more generally areas involving graph modelling with missing edges due to lacking data [17, 33, 37]. Unfortunately minimum completions into most interesting graph classes, *including cographs*, are NP-hard to compute [11, 30, 27, 33, 40]. This fact encouraged researchers to focus on various alternatives that are computationally more efficient, at the cost of optimality or generality. Examples of the approaches that have been attempted include approximation [34], restricted input [7, 6, 32, 29, 10, 28], parameterization [12, 26, 23, 15, 31] and minimal completions [19, 21, 22, 25, 36, 38]. Here we consider the last alternative.

The reason why minimal completions can be used as a tool to understand minimum completions better, is that every minimum completion must also be a minimal one. Hence, if one is able to efficiently sample from the space of minimal completions, it is possible to pick the one in the sample with fewest fill edges and have good chances to produce a completion close to the minimum. This process, while only being a heuristic without any approximation guarantees, has proven to often be good enough for practical purposes [4, 2]. In addition, the study of minimal completions gives a deep insight in the structure of the graph class we consider. It is often the case that new tools created to characterize minimal completions are applied to design new exact algorithms for minimum completions [16, 7, 39], or to efficiently solve other problems on the specific graph class in question. In particular, from a new minimal completion algorithm there can easily follow new recognition algorithms [5, 20], since completion can be regarded as a generalization of recognition. Finally, as shown in [5] for the case of chordal graphs, completions can also be useful to efficiently solve problems that otherwise are hard on the original input graph.

In this paper we consider minimal cograph completions, and we study them both from a graph theoretic and from an algorithmic point of view. Our main graph theoretic result is a theorem that captures the essence of what makes a cograph completion minimal. We apply this characterization to obtain several algorithmic results. First we give a linear time algorithm for the *characterization problem*, that is, for checking whether a given cograph completion is minimal. Second we show how this algorithm can be applied to solve the *extraction problem*, i.e., the problem of extracting a minimal completion from a non-minimal one by removing fill edges. Finally

[*]This work is supported by the Research Council of Norway through grant 166429/V30.

[†]Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Emails: {dlo011, federico, charis}@ii.uib.no

we present our main algorithmic result: an algorithm that solves *computation problem*, namely the problem of compting a minimal cograph completion of an arbitrary input graph. This algorithm can be viewed as a generalization of the cograph recognition algorithm given in [14], due to its incremental nature. We consider, in fact, the input graph one vertex at the time, and we complete it locally in an on-line fashion. Thanks to this feature it is likely that the algorithm can be extended to a dynamic completion algorithm, like the one for split graphs presented in [20]. The running time is linear in the size of the computed minimal cograph completion, and therefore optimal if an explicit representation of the output graph is required.

One should notice that, for cographs, as for other classes for which completions are interesting, an algorithm for the extraction problem can easily be applied to solve the computation problem as well. The reason why we provide a separate algorithm for each problem, is the big difference between their time complexity. While our computation algorithm is linear in output size, the one for the extraction problem runs in time $O(|V(G)|^4)$ in the worst case.

Although we have argued why minimal completions are important in general, we have not yet explained why it is interesting to study minimal *cograph* completions. An obvious reason is that cographs arise naturally in many different fields. It is not by chance that they have been re-discovered various times and have so many different characterizations [13]. Even more interesting is the fact that many problems that are NP-hard on general graphs, can be solved very efficiently when the input is restricted to being a cograph (see [8] for a summary of such results).

However, as noticed by Corneil et. al [14], in most typical applications, the graphs encountered may not be cographs but in fact will be very close to being a cograph. Due to this they asked for good heuristics for the problem of adding and deleting as few edges of the input graph as possible to achieve a cograph. Our computation alorithm can be used as such a heuristic, both in the case of adding and in the case of deleting edges. The reason for this is that the class cographs is self-complementary. Besides, an advantage of using a minimal completion algorithm as a heuristic is that the minimality guarantees that we never add unnecessary fill edges. Also, since our completion algorithm is fast it is possible to improve the performance of the heuristic by trying several different completions and picking the one with fewest edges.

Another reason to study cographs with respect to minimal completions, is that this graph class is not *sandwich monotone* (see [22] for an exact definition). If a graph class has this property, then a completion into the class is minimal if and only if no *single* fill edge can be removed keeping the completed graph in the class. Hence, for polynomial time recognizable classes with this property, it becomes trivial to solve the characterization problem, and very easy to solve both the extraction and the computation problems as well. Examples of algorithms that exploit sandwich monotonicity for efficiently extracting and computing a minimal completion, are those for chordal [4], split [19], threshold and chain graph [22] completions. In contrast, among the classes that do not have the sandwich monotone property, the only one for which a solution to the characterization and extraction probems is known, is the class of interval graphs [24]. When viewed from this perspective, our characterization of minimal cograph completions becomes interesting. It allows us to check minimality efficiently and provides a straightforward way to solve the extraction problem for cograph completions, even though cographs do not have the sandwich monotone property.

Before we begin the technical exposition, we should note that it is possible to adapt the algorithm for the cograph sandwich problem given in [18] to yield a polynomial time algorithm for the extraction problem. However such an algorithm would only be a smart brute force approach and would not give any graph theoretical characterization or intuition on how a minimal cograph completion should look like, which is what we aim for. Also, the running time of the algorithm we would get from such an approach would be too high for any practical purpose.

The paper is organized in three main sections: Section 2 with background and definitions, Section 3 with the details of the characterization and finally, Section 4 with the computation algorithm. The latest section is split in two. The first part contains a high level description of the algorithm for easing the understanding and proving correctness. The second part contains a version more suitable for implementation, together with a running time analysis of the algorithm.

2 Preliminaries

We consider undirected finite graphs with no loops or multiple edges. For a graph $G = (V, E)$, $V(G) = V$ and $E(G) = E$. For $S \subseteq V$, the *subgraph of* G *induced by* S is denoted by $G[S]$. Moreover, we denote by $G - S$ the graph $G[V \setminus S]$ and by $G - v$ the graph $G[V \setminus \{v\}]$. We distinguish between *subgraphs* and *induced subgraphs*. By a *subgraph* of G we mean a graph G' on the same vertex set containing a subset of the edges of G, and we denote it by $G' \subseteq G$. If G' contains a proper subset of the edges of G, we write $G' \subset G$.

The *neighborhood* of a vertex x of G is $N_G(x) = \{v \mid xv \in E\}$. The *degree* of x in G is $d_G(x)$. For $S \subseteq V$ $N_G(S) = \bigcup_{x \in S} N_G(x) \setminus S$. The *complement* $\overline{G}$ of a graph G consists of all vertices and all non-edges of G. A vertex x of G is *universal* if $N_G(x) = V \setminus \{x\}$ and is *isolated* if it has no neighbors in G. A *clique* is a set of pairwise adjacent vertices while an *independent set* is a set of pairwise non-adjacent vertices. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$, their *union* is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Their *join* $G_1 + G_2$ is the graph obtained from $G_1 \cup G_2$ by adding all the edges between the vertices of V_1 and V_2.

A *connected component* of a disconnected graph G is a connected subgraph of G with a maximal set of vertices and edges. The *co-connected components* of G are the connected components of $\overline{G}$. By $\mathcal{C}(G)$ and $\widehat{\mathcal{C}}(G)$ we denote the family of the vertex sets of the connected components and co-connected components, respectively, of G. More formally, $\mathcal{C}(G) = \{C_i \mid G[C_i]$ is a connected component of $G\}$ and $\widehat{\mathcal{C}}(G) = \{\widehat{C}_i \mid G[\widehat{C}_i]$ is a co-connected component of $G\}$.

Given an arbitrary graph $G = (V, E)$ and a graph class Π, a Π *completion* of G is a graph $H = (V, E \cup F)$ such that $H \in \Pi$, and H is a *minimal* Π completion of G if $(V, E \cup F')$ fails to be in Π for every $F' \subset F$. The edges added to the original graph in order to obtain a Π completion are called *fill edges*.

2.1 Cographs

The class of cographs, also known as *complement reducible graphs*, is defined recursively as follows: (i) a single vertex is a cograph, (ii) if G_1 and G_2 are cographs, then $G_1 \cup G_2$ is also a cograph, (iii) if G_1 and G_2 are cographs, then $G_1 + G_2$ is also a cograph. Here we shall use the following characterization of cographs.

Theorem 2.1 ([13]). *G is a cograph if and only if the complement of any nontrivial connected induced subgraph of G is disconnected.*

Along with other properties, it is known that cographs admit a unique tree representation, called a *cotree* [13]. For a cograph G its cotree, denoted by $T(G)$, is a rooted tree having $O(|V|)$ nodes. Notice that we also consider $T()$ as a function that, given a cograph as argument, returns the corresponding cotree. Similarly, we define the function $Co()$, that takes as an input a cotree and returns the corresponding cograph; that is, for a cograph G, $Co(T(G)) = G$. The vertices of G are precisely the leaves of $T(G)$ and every internal node of $T(G)$ is labelled by either 0 (0-node) or 1 (1-node). Two vertices are adjacent in G if and only if their least common ancestor in $T(G)$ is a 1-node. Moreover, if G has at least two vertices then each internal node of the tree has at least two children and any path from the root to any node of the tree consists of alternating 0- and 1-nodes. The complement of any cograph G is a cograph and the cotree of the complement of G is obtained from $T(G)$ with inverted labeling on the internal nodes of $T(G)$. Cographs can be recognized and their cotrees can be computed in linear time [14].

For a node t of $T(G)$ we denote by T_t the subtree rooted at t. The set of t's children in $T(G)$ is denoted by $Q(t)$ and the set of leaves of T_t is denoted by $M(t)$. If $S \subseteq V(T(G))$ then $M(S) = \bigcup_{t \in S} M(t)$. Let $Q(t) = \{t_1, \ldots, t_q\}$. If t is a 0-node then $G[M(t)]$ is disconnected with q connected components and $M(t_i) = C_i$, for $C_i \in \mathcal{C}(G[M(t)])$. Otherwise, if t is a 1-node then $G[M(t)]$ is connected with q co-connected components and $M(t_i) = \widehat{C}_i$, for $\widehat{C}_i \in \widehat{\mathcal{C}}(G[M(t)])$.

Observation 2.2. *Let $G = (V, E)$ be a cograph, $T(G)$ be its cotree, and let $a(G)$ be the set of the 1-nodes of $T(G)$.* $\displaystyle\sum_{t \in a(G)} \sum_{t_i, t_j \in Q(t)} |M(t_i)| \cdot |M(t_j)| = |E|$.

Proof. To prove the statement we use the fact that two vertices are adjacent in G if and only if their least common ancestor (LCA) in $T(G)$ is a 1-node. Therefore we can write that $|E| = \sum_{t \in a(G)} |\{u, v \mid t$ is the

LCA of $u, v\}|$. Since for a given 1-node t, t is the least common ancestor of two vertices x, y if and only if $x \in M(t_i)$ and $y \in M(t_j)$ for distinct $t_i, t_j \in Q(t)$, thus $|\{u, v \mid t \text{ is the } LCA \text{ of } u, v\}| = |M(t_i)| \cdot |M(t_j)|$, and the result follows. □

3 Characterizing minimal cograph completions

Here we exploit certain properties of cographs in order to characterize minimal cograph completions.

Lemma 3.1. *Let $H = (V, E \cup F)$ be a cograph completion of a graph $G = (V, E)$. H is a minimal cograph completion of G if and only if $H[C_i]$ is a minimal cograph completion of $G[C_i]$ for every $C_i \in \mathcal{C}(H)$.*

Proof. Assume $H[C_i]$ is not a minimal cograph completion of $G[C_i]$ for some $C_i \in \mathcal{C}(H)$. Then there exists a graph H' between $G[C_i]$ and $H[C_i]$ that is a cograph and a strict subgraph of $H[C_i]$. Since $H[V \setminus C_i] \cup H'$ is still a cograph and is a strict subgraph of H, H is not minimal. For the other direction, assume that for each $C_i \in \mathcal{C}(H)$, $H[C_i]$ is a minimal cograph completion of $G[C_i]$. Then no subset of the fill edges in each connected component of H can be removed producing a new cograph. Since there are no edges between the connected components of H, it means that no subset of the fill edges can be removed. Hence, H is minimal. □

Let $H = (V, E \cup F)$ be a cograph completion of a graph $G = (V, E)$. Next we focus on a connected cograph completion H of G. Note that H has at least two co-connected components since it is connected. In order to characterize minimality of H, the idea is to consider any two co-connected components of H, remove all the fill edges between them in H, and then simply check the connectivity of the resulting graph. More formally, if $\overline{H}$ is disconnected, let $\widehat{\mathcal{C}}(H) = \{\widehat{C}_1, \ldots, \widehat{C}_\ell\}$. Given two vertex sets $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$, we consider the induced subgraph $H[\widehat{C}_u \cup \widehat{C}_v]$. We build a graph G_{uv} by taking $H[\widehat{C}_u \cup \widehat{C}_v]$ and removing all the fill edges between the two vertex sets $\widehat{C}_u$ and $\widehat{C}_v$. We define

$$G_{uv} = (\widehat{C}_u \cup \widehat{C}_v, E(H[\widehat{C}_u]) \cup E(H[\widehat{C}_v]) \cup E_{uv}),$$

where $E_{uv} = \{xy \mid x \in \widehat{C}_u, y \in \widehat{C}_v, xy \in E\}$. Let us consider now, the subgraphs of H induced by the vertex sets of the connected components of G_{uv}. We define $H_{uv} = \bigcup_{Y_i \in \mathcal{C}(G_{uv})} H[Y_i]$. Notice that if G_{uv} is connected, then $H_{uv} = H[\widehat{C}_u \cup \widehat{C}_v]$; otherwise H_{uv} is disconnected and $G[\widehat{C}_u \cup \widehat{C}_v] \subseteq G_{uv} \subseteq H_{uv} \subset H[\widehat{C}_u \cup \widehat{C}_v]$.

Lemma 3.2. *Let $H = (V, E \cup F)$ be a cograph completion of a graph $G = (V, E)$. H is a minimal cograph completion of G if and only if $H[\widehat{C}_i]$ is a minimal cograph completion of $G[\widehat{C}_i]$, for every $\widehat{C}_i \in \widehat{\mathcal{C}}(H)$ and G_{uv} is a connected graph for any two distinct $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$.*

Proof. Assume that either $H[\widehat{C}_i]$ is not a minimal cograph completion of $G[\widehat{C}_i]$ for some $\widehat{C}_i \in \widehat{\mathcal{C}}(H)$, or G_{uv} is not a connected graph for some distinct $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$. We will show that in both cases $H = (V, E \cup F)$ is not a minimal cograph completion of $G = (V, E)$, because we can build a cograph $H' = (V, E \cup F')$, where $F' \subset F$. In the first case there exists a cograph H'_i which is a strict subgraph of $H[\widehat{C}_i]$. Therefore we can define the cograph $H' = H[V \setminus \widehat{C}_i] + H'_i$, that is clearly a strict subgraph of H. For the second case assume that for every $\widehat{C}_i \in \widehat{\mathcal{C}}(H)$, $H[\widehat{C}_i]$ is a minimal cograph completion of $G[\widehat{C}_i]$, but for at least two distinct $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$, G_{uv} is not connected. Since G_{uv} is not connected, the cograph H_{uv} is a strict subgraph of $H[\widehat{C}_u \cup \widehat{C}_v]$. Thus the graph $H' = H[V \setminus (\widehat{C}_u \cup \widehat{C}_v)] + H_{uv}$ is a cograph by Theorem 2.1 and by construction we know that $G \subseteq H' \subset H$. Hence H is not minimal.

To prove the other direction we show that if $H[\widehat{C}_i]$ is a minimal cograph completion of $G[\widehat{C}_i]$ for each $\widehat{C}_i \in \widehat{\mathcal{C}}(H)$, and G_{uv} is a connected graph for each two distinct $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$, then H is a minimal cograph completion of G. Since $H[\widehat{C}_i]$ is a minimal cograph completion of $G[\widehat{C}_i]$, no fill edge can be removed from these subgraphs. Assume for the sake of contradiction that H is not minimal. Then there must exist a cograph H' such that $G \subseteq H' \subset H$. By assumption we know that $H'[\widehat{C}_i] = H[\widehat{C}_i]$ for each $\widehat{C}_i \in \widehat{\mathcal{C}}(H)$, and $H'[\widehat{C}_u \cup \widehat{C}_v]$ is connected as a supergraph of G_{uv}, for any two distinct $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$. We show that H' cannot be a cograph, contradicting the existence of H'. Since $H' \subset H$, there is at least a non-edge in H' between two vertex sets

$\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$, so that $H'[\widehat{C}_u]$ is not universal for $H'[\widehat{C}_v]$. This means that $\overline{H'}[\widehat{C}_u \cup \widehat{C}_v]$ is a connected graph because both $\overline{H'}[\widehat{C}_u]$ and $\overline{H'}[\widehat{C}_v]$ are connected graphs (since they are connected components in $\overline{H}$) and there is at least one edge between them. Hence, both $H'[\widehat{C}_u \cup \widehat{C}_v]$ and $\overline{H'}[\widehat{C}_u \cup \widehat{C}_v]$ are connected and H' cannot be a cograph by Theorem 2.1. □

As we are about to see, in order to check the connectivity of G_{uv} it is enough to consider only the edges between the co-connected components $\widehat{C}_u$ and $\widehat{C}_v$ and not the ones that are inside $\widehat{C}_u$ and $\widehat{C}_v$. For that reason we introduce the graph G^*_{uv} which can be viewed as the graph obtained from G_{uv} by replacing every connected component of $H[\widehat{C}_u]$ and $H[\widehat{C}_v]$ with a single vertex.

We formally define the graph G^*_{uv} over the cotree $T(H)$ of H. Let t_u, t_v be two children of the root of $T(H)$. If $Q(t_u) \neq \emptyset$ then let $A_u = Q(t_u)$; otherwise let $A_u = \{t_u\}$. Similarly if $Q(t_v) \neq \emptyset$ then let $A_v = Q(t_v)$; otherwise let $A_v = \{t_v\}$. Observe that A_u and A_v contain nodes of $T(H)$. Given the two nodes t_u, t_v we define the graph G^*_{uv} as follows.

$$G^*_{uv} = (A_u \cup A_v, E^*_{uv}),$$

where $E^*_{uv} = \{(a_u, a_v) \in A_u \times A_v \mid (M(a_u) \times M(a_v)) \cap E \neq \emptyset\}$. In other words, edges are between A_u and A_v, and a vertex a_u of A_u is adjacent to a vertex a_v of A_v if and only if there is an edge $xy \in E$ such that $x \in M(a_u)$ and $y \in M(a_v)$. This also means that G^*_{uv} is a bipartite graph.

An example of the graphs G_{uv} and G^*_{uv} is given in Figure 1.

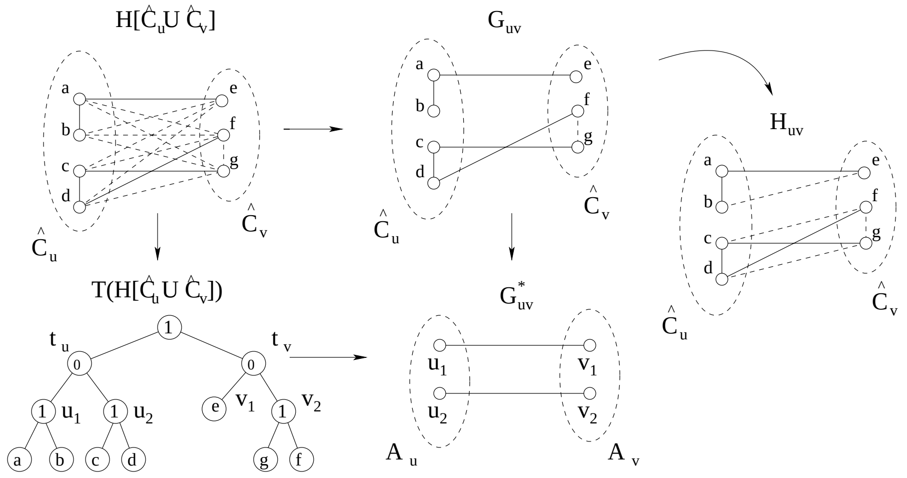

Figure 1: An example of the graphs G_{uv}, G^*_{uv} and H_{uv}. The dashed lines represent fill edges added to the original graph. We depict the way that G_{uv} is obtained from $H[\widehat{C}_u \cup \widehat{C}_v]$ by removing the fill edges between $H[\widehat{C}_u]$ and $H[\widehat{C}_v]$. Then we can think of G^*_{uv} as either the graph obtained from G_{uv} by contracting the connected component in each side, or directly defined on the cotree of $H[\widehat{C}_u \cup \widehat{C}_v]$ when we do not consider fill edges between the children of t_u and the children of t_v.

Observation 3.3. *Let $T(H)$ be the cotree of a connected cograph completion H of G and let t be the root of $T(H)$ and $t_u, t_v \in Q(t)$. G^*_{uv} is connected if and only if G_{uv} is connected, where $\widehat{C}_u = M(t_u)$ and $\widehat{C}_v = M(t_v)$. Moreover, for any element $Y_i \in \mathcal{C}(G^*_{uv})$, $M(Y_i) \in \mathcal{C}(G_{uv})$.*

Proof. Let us explain the graph G^*_{uv} with respect to G_{uv}. Given two distinct sets $\widehat{C}_u, \widehat{C}_v \in \widehat{\mathcal{C}}(H)$, notice that $G_{uv}[\widehat{C}_u] = H[\widehat{C}_u]$ and $G_{uv}[\widehat{C}_v] = H[\widehat{C}_v]$ are disconnected cographs or a single vertex, since $H[M(t)]$ is connected. Thus every connected component of $G_{uv}[\widehat{C}_u]$ and $G_{uv}[\widehat{C}_v]$ corresponds to a vertex of A_u and A_v, respectively of G^*_{uv}. Moreover there is at least one edge between two connected components of $G_{uv}[\widehat{C}_u]$

and $G_{uv}[\widehat{C}_v]$ if and only if there is an edge between the corresponding vertices of G^*_{uv}. Hence the statement follows. □

We can now rephrase Lemma 3.2 in terms of the cotree of H instead of H itself, and using G^*_{uv} instead of G_{uv}.

Theorem 3.4. *Let H be a cograph completion of a graph G and let $T(H)$ be its cotree. H is a minimal cograph completion of G if and only if for every 1-node t of $T(H)$ the graph G^*_{uv} is connected for any two nodes $t_u, t_v \in Q(t)$.*

Proof. Suppose H is a minimal cograph completion of G. Consider a 1-node t of $T(H)$, and let t_u,$t_v \in Q(t)$. The subtree of $T(H)$ rooted at t is the cotree of $H[M(t)]$. Furthermore $H[M(t)]$ is a minimal connected cograph completion of $G[M(t)]$, $M(t_u) \in \widehat{\mathcal{C}}(H[M(t)])$ and $M(t_v) \in \widehat{\mathcal{C}}(H[M(t)])$. Now, by Lemma 3.2 and Observation 3.3 G^*_{uv} is connected.

We prove the other direction of the equivalence by induction on $|V(T(H))|$. If $|V(T(H))| = 1$ then G has only one vertex and the result follows. Assume that the statement of the theorem holds whenever $|V(T(H))| < k$. Consider now the case when $|V(T(H))| = k$. By the induction hypothesis $H[M(t_u)]$ is a minimal cograph completion of $G[M(t_u)]$ for every $t_u \in Q(root(T(H)))$. Thus, if $root(T(H)))$ is a 0-node then the result follows by Lemma 3.1. Furthermore if $root(T(H)))$ is a 1-node then we know that G^*_{uv} is connected for every pair of children of the root. Therefore H is a minimal cograph completion of G by Lemma 3.2 and Observation 3.3. □

Based on the previous theorem we obtain a linear-time algorithm for deciding whether a given cograph completion is minimal.

Theorem 3.5. *Let $H = (V, E \cup F)$ be a cograph completion of a graph $G = (V, E)$. Recognizing whether H is a minimal cograph completion of G can be done in $O(|V| + |E| + |F|)$ time.*

Proof. We describe such an algorithm. First we compute the cotree $T(H)$ of H. Then we visit each 1-node t of $T(H)$. Let $Q(t) = \{t_1, \ldots, t_\ell\}$. For every pair of nodes $(t_u, t_v) \in Q(t)$ we construct the graph G^*_{uv} and check its connectivity. If at least one of the graphs G^*_{uv} is disconnected then we output that H is not a minimal cograph completion of G; otherwise we output that H is a minimal cograph completion of G. The correctness follows by Theorem 3.4.

Let us now show that the algorithm runs in linear time. Observe that the cotree can be computed in time linear in the size of H and has $O(|V|)$ nodes [14]. Let t be a 1-node in $T(H)$ and let $Q(t) = \{t_1, \ldots, t_\ell\}$. We need to construct the graph G^*_{uv} for each pair of nodes $(t_u, t_v) \in Q(t)$. Let $n_u = |M(t_u)|$ and $n_v = |M(t_v)|$. Note that the subtrees T_{t_u} and T_{t_v} have $O(n_u)$ and $O(n_v)$ nodes, respectively. Thus finding the sets $M(t_u)$ and $M(t_v)$ takes time $O(n_u + n_v)$. For the edges of G^*_{uv} we do not need to check any edge inside $H[M(t_u)]$ and $H[M(t_v)]$, but only the edges in between. This implies that building and checking the connectivity of G^*_{uv} take time $O(n_u n_v)$ using an adjacency matrix, since $n_u + n_v \leq n_u n_v + 1$ (note that using a trick in [1], the matrix can be allocated in linear time). Observe that t is a 1-node meaning that there are $O(n_u n_v)$ edges in $H[M(t)]$. Thus summing up the time needed for each distinct pair of nodes (t_u, t_v) in $T(G)$, gives time linear in the size of H by Observation 2.2. Therefore the overall running time is $O(|V| + |E| + |F|)$. □

Using the previous theorem, we can give an algorithm for extracting a minimal cograph completion from a given one. The idea is quite simple. On input G and H we use the algorithm from Theorem 3.5 to check whether H is a minimal cograph completion of G. If the answer is yes we can output H, while if the answer is no, there must be a 1-node t of $T(H)$ with children u and v such that the graph G^*_{uv} is disconnected. In that case H_{uv} is a cograph completion of $G[M(u) \cup M(v)]$ such that $H_{uv} \subset H[M(u) \cup M(v)]$. Thus $H' = (H \setminus (M(u) \cup M(v))) + H_{uv})$ is a cograph completion of G such that $H' \subset H$. We can now reiterate this process with H' as a candidate cograph completion. Since each such iteration can be done in $O(|V|+|E|+|F|)$ time and we remove at least one fill edge for each iteration, this algorithm runs in $O((|V|+|E|+|F|)|F|)$ time. One should notice that our extraction algorithm has many similarities with the generic extraction algorithm for sandwich monotone graph classes [22]. Similarly to sandwich monotonicity, our characterization states that in some sense, a cograph completion is minimal if and only if it is *locally* minimal.

Theorem 3.6. *Given a cograph completion $H = (V, E \cup F)$ of a graph $G = (V, E)$, a minimal cograph completion H' with $G \subseteq H' \subseteq H$, can be computed in time $O((|V| + |E| + |F|) \cdot |F|)$.*

4 Computing a minimal cograph completion directly

In this section we give an algorithm to solve the problem of computing a minimal cograph completion of an arbitrary input graph, in time linear in the size of the output graph. We use a vertex incremental scheme proposed in [21] for computing minimal cograph completions.

It is known that given a graph class that is hereditary[1] and has the universal vertex property[2], minimal completions of arbitrary graphs into this class can be computed in a vertex incremental way [3, 21]. Cographs satisfy both properties by Theorem 2.1 and thus we state this result in the following lemma.

Proposition 4.1. *Let H be a minimal cograph completion of an arbitrary graph G, and let G_x be a graph obtained from G by adding a new vertex x adjacent to some vertices of G. There is a minimal cograph completion H_x of G_x such that $H_x - x = H$.*

At all times we maintain a minimal cograph completion of the part of the input graph that already has been considered. The algorithm starts off with the empty graph and adds in the vertices of the input graph one at a time, at each step updating the minimal completion by adding fill edges incident to the new vertex. The main technical part of this section is the design and analysis of an algorithm for one incremental step.

4.1 Adding a vertex to a cograph

In this section we give an algorithm for one incremental step of our completion algorithm. Hereafter we use $G = (V, E)$ to denote a cograph, unless otherwise specified. Given a vertex x together with a list of vertices $N_x \subseteq V$, we denote by G_x the graph obtained by adding x to G. That is, $G_x = (V \cup \{x\}, E \cup \{xy : y \in N_x\})$. Given a cograph G and a vertex set $N_x \subseteq V$ the algorithm computes a vertex set $S \subseteq V$ such that $N_x \subseteq S$ and $H_x = (V \cup \{x\}, E \cup \{xy : y \in S\})$ is a minimal cograph completion of G_x.

The algorithm is fairly simple. We start off with G_x and consider $G = G_x - x$. If G is disconnected we only need to add edges to the connected components of G that x is already adjacent to. If x is adjacent to only one connected component we run the algorithm recursively on that connected component. However, if x is adjacent to more than one connected component of G we make x universal to all connected components of G that are adjacent to x. When G is connected we have to be a bit more careful. The basic idea is the following: We try adding x to a particular co-connected component $\widehat{C}$. To do this we have to make x universal to all other co-connected components of G and make sure not to make x universal to $\widehat{C}$. If we find out that x cannot be added to any co-connected component in this way, we make x universal to all co-connected components of G that x is adjacent to in G_x. In order to justify these choices, we will apply Theorem 3.1.

[1]A graph class Π is called *hereditary* if all induced subgraphs of graphs in Π also belong to Π.

[2]A graph class Π has the *universal vertex property* if, for every graph $G \in \Pi$ and a vertex $x \notin V$, $G + x \in \Pi$.

```
Algorithm: Minimal_x_Cograph_Completion – MxCC ($G$, $N_x$ )
Input: A cograph $G$, and a set of vertices $N_x$ which are to be made adjacent to a vertex $x \notin V$
Output: An inclusion minimal set $S \subseteq V$ such that $N_x \subseteq S$ and $H_x = (V \cup \{x\}, E \cup \{xy : y \in S\})$
        is a cograph
if $G$ is connected then
    if there are $\widehat{C}_i \in \widehat{\mathcal{C}}(G)$ and $C_j \in \mathcal{C}(G[\widehat{C}_i])$ s.t. $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$ then
        $S = MxCC(G[\widehat{C}_i], N_x \cap \widehat{C}_i) \cup (V \setminus \widehat{C}_i)$;
    else
        $S = \bigcup_{\widehat{C}_i \in \widehat{\mathcal{C}}(G)\,:\,\widehat{C}_i \cap N_x \neq \emptyset} \widehat{C}_i$;
else
    if there is a $C_i \in \mathcal{C}(G)$ such that $N_x \subseteq C_i$ then
        $S = MxCC(G[C_i], N_x)$;
    else
        $S = \bigcup_{C_i \in \mathcal{C}(G)\,:\,C_i \cap N_x \neq \emptyset} C_i$;
return $S$;
```

Observe that the algorithm always terminates because each recursive call takes as an argument a subgraph of G induced by a strict subset of V. We are now ready to prove the correctness of Algorithm $MxCC$.

Lemma 4.2. *Given a cograph G and a set of vertices N_x, Algorithm $MxCC$ returns a set of vertices S such that H_x is a cograph completion of G_x.*

Proof. Observe that as $N_x \subseteq S$ we know that $G_x \subseteq H_x$. Thus it is sufficient to show that H_x is a cograph. We prove this by induction on $|V|$. If $|V| \leq 1$ then H_x has at most two vertices and is a cograph. Assume now that the statement of the lemma holds whenever the input graph has less than k vertices and consider the execution of Algorithm $MxCC(G, N_x)$ on a graph G on k vertices. For each of the following cases we will prove that H_x can be constructed by either taking the union or the join of two cographs. This implies that H_x is a cograph by Theorem 2.1.

First we consider the case when G is connected. If $S = MxCC(G[\widehat{C}_i], N_x \cap \widehat{C}_i) \cup (V \setminus \widehat{C}_i)$ then let S' be the set returned by $MxCC(G[\widehat{C}_i], N_x \cap \widehat{C}_i)$ and let $H'_x = (\widehat{C}_i \cup \{x\}, E(G[\widehat{C}_i]) \cup \{xy : y \in S'\})$. By the induction hypothesis H'_x is a cograph. Thus $H_x = H'_x + G[V \setminus \widehat{C}_i]$ is a cograph. If $S = \bigcup_{\widehat{C}_i \in \widehat{\mathcal{C}}(G):\widehat{C}_i \cap N_x \neq \emptyset} \widehat{C}_i$ then $H_x = (G[V \setminus S] \cup \{x\}) + G[S]$ is a cograph.

Now consider the case when G is disconnected. If there is a $C_i \in \mathcal{C}(G)$ such that $C_i \subseteq N_x$, then let S' be the set returned by $MxCC(G[C_i], N_x)$ and $H'_x = (C_i \cup \{x\}, E(G[C_i]) \cup \{xy : y \in S'\})$. By the induction hypothesis H'_x is a cograph. Thus $H_x = H'_x \cup G[V \setminus C_i]$ is a cograph. If $S = \bigcup_{C_i \in \mathcal{C}(G)\,:\,C_i \cap N_x \neq \emptyset} C_i$ then $H_x = (G[S] + x) \cup G[V \setminus S]$ is a cograph. □

Observation 4.3. *If G is disconnected, $C_i \in \mathcal{C}(G)$, $N_x \cap C_i = \emptyset$, and $S = MxCC(G, N_x)$ then $S \cap C_i = \emptyset$.*

Proof. If $N_x \subseteq C_j$ for a $C_j \in \mathcal{C}(G)$ and $C_j \neq C_i$ then the claim follows immediately as $S \subseteq C_j$. If $C_i = C_j$ then the call to $MxCC(G[C_j], N_x)$ returns an empty set as $N_x = \emptyset$. Finally, if there is no $C_j \in \widehat{\mathcal{C}}(G)$ so that $N_x \subseteq C_j$, then $S = \bigcup_{C_k \in \mathcal{C}(G)\,:\,C_k \cap N_x \neq \emptyset} C_k$ and the result follows. □

Theorem 4.4. *Given a cograph G and a set of vertices N_x, Algorithm $MxCC$ returns a set of vertices S such that H_x is a minimal cograph completion of G_x.*

Proof. By Lemma 4.2, H_x is a cograph completion of G_x. Thus it is sufficient to show minimality. We prove that H_x is a minimal cograph completion of G_x by induction on $|V|$. If $|V| \leq 1$, H_x is trivially a minimal completion of G_x. Now, assume that the statement of the theorem holds whenever the input graph has less than k vertices and consider the execution of Algorithm $MxCC(G, N_x)$ on a graph G on k vertices. We distinguish between two cases according to the connectivity of G and prove that at each case H_x is a minimal cograph completion of G_x by using Lemmas 3.1 and 3.2.

First we consider the case when G is connected. If there are $\widehat{C}_i \in \widehat{\mathcal{C}}(G)$ and $C_j \in \mathcal{C}(G[\widehat{C}_i])$ such that $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$, then let S' be the set returned by $MxCC(G[\widehat{C}_i], N_x \cap \widehat{C}_i)$. Given the set S', consider the graphs $G'_x = (\widehat{C}_i \cup \{x\}, E(G[\widehat{C}_i]) \cup \{xy : y \in \widehat{C}_i \cap N_x\})$ and $H'_x = (\widehat{C}_i \cup \{x\}, E(H[\widehat{C}_i]) \cup \{xy : y \in S'\})$. Now, since $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$ we know that $C_j \subset \widehat{C}_i$, $G[\widehat{C}_i]$ is disconnected, and $G[C_j]$ is a connected component of $G[\widehat{C}_i]$. Thus, by Observation 4.3, $C_j \cap S' = \emptyset$. From this it follows that in H_x, x has a non-neighbour in $\widehat{C}_i$. Furthermore, as $G[\widehat{C}_i]$ is a co-connected component of G and x is universal to $V \setminus \widehat{C}_i$ in H_x we conclude that $H[\widehat{C}_i \cup \{x\}]$ is a co-connected component of H_x. Finally, as G is connected and N_x is nonempty, H_x is connected.

We wish to apply Lemma 3.2 in order to show that H_x is a minimal cograph completion of G_x. Let $\widehat{C}_u$ and $\widehat{C}_v$ be the vertex sets of two distinct co-connected components of H_x. If neither $\widehat{C}_u$ nor $\widehat{C}_v$ contains x we know that both $G[\widehat{C}_u]$ and $G[\widehat{C}_v]$ are co-connected components of G. Thus G_{uv} is just $G[C_u \cup C_v]$, so G_{uv} is connected. Now, without loss of generality, $\widehat{C}_u$ contains x. By the discussion in the paragraph above, $\widehat{C}_u = \widehat{C}_i \cup \{x\} = V(H'_x)$. By the induction hypothesis $H_x[\widehat{C}_u]$ is a minimal cograph completion of $G_x[\widehat{C}_u]$. Again $H_x[\widehat{C}_v] = G_x[\widehat{C}_v]$. We now proceed to show the connectivity of G_{uv}. Obviously, $G[\widehat{C}_u \cup \widehat{C}_v \setminus \{x\}] = G[\widehat{C}_i \cup \widehat{C}_v] \subseteq (G_{uv} - x)$. Additionally, as $\widehat{C}_i$ and $\widehat{C}_v$ are vertex sets of co-connected components of G, $G[\widehat{C}_i \cup \widehat{C}_v]$ is connected. As $G[\widehat{C}_i \cup \widehat{C}_v] \subseteq (G_{uv} - x)$, $(G_{uv} - x)$ is connected. But since $\widehat{C}_i \cap N_x \neq \emptyset$, x has a neighbour in C_i so G_{uv} is connected as well. Therefore H_x is a minimal cograph completion of G_x by Lemma 3.2.

Now suppose that there are no $\widehat{C}_i \in \mathcal{C}(\overline{G})$ and $C_j \in \mathcal{C}(G[\widehat{C}_i])$ such that $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$. If $N_x = \emptyset$ then $S = \emptyset$ and H_x is trivially a minimal completion of G_x. Otherwise, let us describe the co-connected components of H_x. Let $\widehat{C}_1, ..., \widehat{C}_n$ be the elements of $\widehat{\mathcal{C}}(G)$ such that $\widehat{C}_s \cap N_x \neq \emptyset$ for $1 \leq s \leq n$, and let $\widehat{C}'_{n+1} = \bigcup_{\widehat{C} \in \widehat{\mathcal{C}}(G) : \widehat{C} \cap N_x = \emptyset} \widehat{C}$. In H_x, x is adjacent to every vertex of $\widehat{C}_1, ..., \widehat{C}_n$ and isolated to $\widehat{C}'_{n+1}$. Thus H_x has $n+1$ co-connected components, induced by the sets $\widehat{C}_1, ..., \widehat{C}_n$ and $\{x\} \cup \widehat{C}'_{n+1}$. Observe that $H_x[\{x\} \cup \widehat{C}'_{n+1}] = G_x[\{x\} \cup \widehat{C}'_{n+1}]$ and thus $H_x[\{x\} \cup \widehat{C}'_{n+1}]$ is a minimal cograph completion of $G_x[\{x\} \cup \widehat{C}'_{n+1}]$. The same holds for $H_x[\widehat{C}_s]$, $1 \leq s \leq n$, since x is not contained in $\widehat{C}_s$. Furthermore, for any two distinct integers u and v between 1 and n, the graph G_{uv} is just $G[\widehat{C}_u \cup \widehat{C}_u]$ and is connected. To complete the proof, we prove that the graph obtained from $H_x[\{x\} \cup \widehat{C}'_{n+1}]$, $H_x[\widehat{C}_s]$ and by adding the edges of G_x in between, namely G_{uv}, is connected. The minimality then follows by Lemma 3.2. Notice that every vertex of $\widehat{C}'_{n+1}$ is adjacent to every vertex of $\widehat{C}_s$ as a vertex set of a co-connected component of G. What remains to show is that $G_x[\{x\} \cup \widehat{C}_s]$ is connected for any s between 1 and n. Indeed $G_x[\widehat{C}_s] = G[\widehat{C}_s]$ is disconnected but x is adjacent to at least one vertex of each of $G[\widehat{C}_s]$'s connected components, since by construction $\widehat{C}_s \cap N_x$ is nonempty, and thus by assumption $C_j \cap N_x$ is nonempty for every $C_j \in \mathcal{C}(G[\widehat{C}_s])$. We conclude that H_x is a minimal cograph completion of G_x by Lemma 3.2.

Next we consider the case when G is disconnected. If there is a $C_i \in \mathcal{C}(G)$ such that $N_x \subseteq C_i$ then H_x is a disconnected cograph and $H_x[C_i \cup \{x\}]$ is a minimal cograph completion of $G_x[C_i \cup \{x\}]$ by the induction hypothesis. Moreover for every other $C_j \in \mathcal{C}(G)$ such that $C_j \neq C_i$, $H_x[C_j]$ is a connected component of H_x and $H_x[C_j] = G_x[C_j]$. Thus by Lemma 3.1 H_x is a minimal cograph completion of G_x.

Otherwise, let H'_x be the connected component of H_x containing x, and let $G'_x = G_x[V(H'_x)]$. By Lemma 3.1, it is enough to prove that H'_x is a minimal cograph completion of G'_x to show the minimality of H_x. By construction $H'_x = G[\bigcup_{C_i \in \mathcal{C}(G) \,:\, C_i \cap N_x \neq \emptyset} C_i] + x$. Also, by assumption, $|C_i \in \mathcal{C}(G) \,:\, C_i \cap N_x \neq \emptyset| \geq 2$. Thus, H'_x has two co-connected components: $H'_x[\{x\}]$ and $H'_x - x$. Let $C_u = \{x\}$ and $C_v = V(H'_x) \setminus \{x\}$. By definition $G_{uv} = G'_x$, and since x has neighbors in each connected component of $H'_x[C_v] = G'_x[C_v]$, it is easy to see that G_{uv} is connected. Hence the result follows by Lemma 3.2. □

4.2 Implementing Algorithm *MxCC* using a cotree representation

In order to obtain a good running time for Algorithm $MxCC$ we give an algorithm that works directly on the cotree of the input graph. That is, we give an algorithm, namely $CMxCC$, that takes as an input the cotree $T(G)$ of a cograph G and a set N_x of vertices in G and returns a set of vertices S of G so that H_x is a minimal cograph completion of G_x. For a node t in $T(G)$, recall that $Q(t)$ is the set of t's children in $T(G)$. Let $Q_x(t) = \{t_i \in Q(t) : M(t_i) \cap N_x \neq \emptyset\}$. Thus $Q_x(t) \subseteq Q(t)$.

```
Algorithm: Cotree_Minimal_x_Cograph_Completion – CMxCC ( $T$, $N_x$ )
Input: A cotree $T$ of a cograph $G = (V, E)$ and a set of vertices $N_x$ which are to be made adjacent to a
        vertex $x \notin V$
Output: An inclusion minimal set $S \subseteq V$ such that $N_x \subseteq S$ and $H_x = (V \cup \{x\}, E \cup \{xy : y \in S\})$ is
        a cograph
$r = root(T)$ ;
if $r$ is a 1-node then
    if there is a $t \in Q_x(r)$ such that $\emptyset \subset Q_x(t) \subset Q(t)$ then
        $S = CMxCC(T_t, N_x \cap M(t)) \cup (M(r) \setminus M(t))$;
    else
        $S = \bigcup_{t \in Q_x(r)} M(t)$;
else
    if there is a $t \in Q(r)$ such that $Q_x(r) \subseteq \{t\}$ then
        $S = CMxCC(T_t, N_x)$;
    else
        $S = \bigcup_{t \in Q_x(r)} M(t)$;
return $S$;
```

The correctness of the algorithm follows from the following two observations which imply that Algorithm $CMxCC$ returns the same set S as Algorithm $MxCC$.

Observation 4.5. *Let G be a connected cograph and let r be the root of $T(G)$. There are vertex sets $\widehat{C}_i \in \widehat{\mathcal{C}}(G)$ and $C_j \in \mathcal{C}(G[\widehat{C}_i])$ such that $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$ if and only if there is a node $t \in Q_x(r)$ such that $\emptyset \subset Q_x(t) \subset Q(t) \neq \emptyset$.*

Proof. In one direction, suppose r has a child t in $Q_x(r)$ such that $\emptyset \subset Q_x(t) \subset Q(t) \neq \emptyset$. Then $Q_x(t) \setminus Q(t)$ Let $\widehat{C}_i = M(r)$, and let c be an element of $Q(r) \setminus Q_x(r)$. By construction, $G[\widehat{C}_i]$ is a co-connected component of G, $G[C_j]$ is a connected component of $G[\widehat{C}_i]$, $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$. In the other direction, suppose there are vertex sets $\widehat{C}_i \in \widehat{\mathcal{C}}(G)$ and $C_j \in \mathcal{C}(G[\widehat{C}_i])$ such that $\widehat{C}_i \cap N_x \neq \emptyset$ and $C_j \cap N_x = \emptyset$. As $\widehat{C}_i \in \widehat{\mathcal{C}}(G)$, r has a child t such that $\widehat{C}_i = M(t)$. Furthermore, as there is a $C_j \in \mathcal{C}(G[\widehat{C}_i])$ such that $\widehat{C}_j \cap N_x = \emptyset$, t has a child c with $M(c) = C_j$ and thus $c \in Q(t) \setminus Q_x(t)$ and thus $Q_x(t) \subset Q(t)$. As $M(t) \cap N_x \neq \emptyset$ it follows that $t \in Q_x(r)$ and $\emptyset \subset Q_x(t)$. □

Observation 4.6. *Let G be a disconnected cograph and let r be the root of $T(G)$. There is a set $C_i \in \mathcal{C}(G)$ such that $N_x \subseteq C_i$ if and only if there is a node $t \in Q(r)$ such that $Q_x(r) \subseteq \{t\}$.*

Proof. Suppose there is a set $C_i \in \mathcal{C}(G)$ such that $N_x \subseteq C_i$. Then, let t be the child of r so that $M(t) = C_i$. For any other child t' of r, clearly $M(t') = \emptyset$. Thus $Q_x(r) \subseteq \{t\}$. In the other direction, suppose $Q_x(r) \subseteq \{t\}$ for some child t of r. Let $C_i = M(t)$. We know that $C_i \in \mathcal{C}(G)$. We prove that for every $C_j \in \mathcal{C}(G)$ so that $C_j \neq C_i$, $C_j \cap N_x = \emptyset$. Suppose for contradiction that $C_j \cap N_x \neq \emptyset$. Let c be the child of r so that $M(c) = C_j$. Clearly $c \neq t$ and $c \in Q_x(r)$ contradicting that $Q_x(r) \subseteq \{t\}$. □

Now we are ready to prove a bound on the running time for computing a minimal cograph completion H_x of G_x, and give the final theorem about the existence of an algorithm to compute a minimal cograph completion in time linear in the size of the output graph.

Theorem 4.7. *Given a cograph G and its cotree $T(G)$, there is an algorithm for computing the set of vertices S that are adjacent to x in a minimal cograph completion of G_x which runs in $O(|S| + 1)$ time.*

Proof. We describe such an algorithm. First we compute the set $Q_x(t)$ for every node t in $T(G)$ and then we apply Algorithm $CMxCC$ on $T(G)$. By the previous arguments and Theorem 4.4 the set S returned by Algorithm $CMxCC$ contains the vertices that are adjacent to x in a minimal cograph completion H_x of G_x. Now we analyze the running time.

In addition to the work described below the algorithm does a constant amount of work. This does not pose any problem if $S \neq \emptyset$. However if $S = \emptyset$ then we need to add a constant to the running time bound to cover this case. If $S = \emptyset$ it is easy to see that the algorithm requires constant time. Thus, in the rest of the proof we will assume that $S \neq \emptyset$.

Let us show first that given the set $Q_x(t)$ for every node t in $T(G)$, Algorithm $CMxCC$ makes $O(|S|)$ calls. Let $R = \{r_1, r_2, \dots, r_\ell\}$ where r_i be the root of the subtree considered at the ith call of Algorithm $CMxCC$. Now we show that $\ell = O(|S|)$. The nodes of R form a path in $T(G)$ starting from the root ($= r_1$), since at most one child of an internal node of $T(G)$ is given as an argument in each call of $CMxCC$. Observe that the labels of the internal nodes (0- or 1-nodes) of a cotree $T(G)$ alternate along any path starting at the root. Thus at least $\lfloor \frac{\ell}{2} \rfloor$ nodes of R are 1-nodes. For the 1-nodes the algorithm adds at least one vertex of G in S. Hence $\ell = O(|S|)$.

Next we prove that we can compute the set $Q_x(t)$ for every node t in $O(|S|)$ time. Before executing Algorithm $CMxCC$ we start from the leaves of $T(G)$ which correspond to the neighbors of x and then in a bottom up fashion we construct the set P of all internal nodes t in $T(G)$ that satisfy $|Q_x(t)| > 0$. We need to show that $|P| = O(|S|)$. Observe that $R \subseteq P$, since $r_{i+1} \in Q_x(r_i)$ where $r_i, r_{i+1} \in R$. Recall that $\ell = |R|$ and $\ell = O(|S|)$. Thus we need to show that $|P \setminus R| = O(|S|)$. For a node t in $P \setminus R$ observe that $M(t) \subseteq S$ since by the algorithm $\bigcup_{t \in Q_x(r)} M(t) \subseteq S$ where $r \in R$. Therefore $|P \setminus R| = O(|S|)$.

Having computed the set $Q_x(t)$ for each node t in $T(G)$ we show that checking the conditions in the if statements can be done in constant time. In order to do this we traverse the nodes of P once more. Let t be a node in P and let r be t's parent in $T(G)$. Note that if $|Q_x(t)| > 0$ then $t \in Q_x(r)$. If t is a 0-node and $0 < |Q_x(t)| < |Q(t)|$ then r marks its child t, unless r already has a marked child. If r already has a marked child we do nothing.

If r is a 1-node we now can test whether there is a $t \in Q_x(r)$ such that $\emptyset \subset Q_x(t) \subset Q(t)$ in constant time, simply by checking whether r has a marked child or not. Moreover if r is a 0-node, we can test whether there is a $t \in Q(r)$ such that $Q_x(r) \subseteq \{t\}$ in constant time, by checking that $|Q_x(r)| \leq 1$. This implies that the algorithm terminates in $O(|S| + 1)$ time. □

Theorem 4.8. *There is an algorithm for computing a minimal cograph completion $H = (V, E \cup F)$ of an arbitrary graph $G = (V, E)$ in $O(|V| + |E| + |F|)$ time.*

Proof. Let $n = |V|$. Order the vertices of G from v_1 to v_n, let $V_i = \{v_1, v_2, \dots v_i\}$ and $G_i = G[V_i]$. Let $H_1 = G_1$ and $S_{i+1} = CMxCC(T_i, N_{G_{i+1}}(v_{i+1}))$ where T_i is the cotree of H_i. Construct H_{i+1} from H_i by adding the vertex v_{i+1} and making v_{i+1} adjacent to S_{i+1}. Obviously, T_1 is the cotree of a minimal cograph completion of G_1. If T_i is the cotree of a minimal cograph completion H_i of G_i, then Theorem 4.4 yields that T_{i+1} is the cotree of a minimal cograph completion H_{i+1} of G_{i+1}. Thus, by induction, T_n is the cotree of a minimal cograph completion $H_n = H$ of $G_n = G$. Finally, we consider the running time for computing H by using the adjacency list of G. Computing S_{i+1} from T_i takes $O(|S_{i+1}|+1)$ time by Theorem 4.7, where $S_{i+1} = N_{H_{i+1}}(v_{i+1})$. Note also that T_{i+1} can be computed directly from T_i and S_{i+1} in $O(|S_{i+1}|)$ time since updating the cotree requires $O(d)$ time whenever the addition of a vertex of degree d results in a cograph [14]. Therefore the total running time becomes $\sum_{i=2}^{n} O(|S_i| + 1) = O(|V|) + O(\sum_{i=2}^{n} d_H(v_i)) = O(|V| + |E| + |F|)$. □

5 Concluding remarks

We have studied minimal cograph completions from two different points of view. Our results include an efficient algorithm for the computation problem and a precise characterization of minimal cograph completions. Such characterizations are rarely known for graph classes that do not have the sandwich monotone property. Observe, in fact, that for comparability and proper interval graphs, there exists a computation algorithm [21, 36] while no characterization of minimal completions is known. This makes our characterization and the consequent extraction algorithm particularly interesting.

Three interesting problems we leave open are:

1. *Can one design a faster extraction algorithm, possibly linear in the size of the given completion?*
2. *Does there exist a computation algorithm running in time linear in the size of the input graph?*
3. *Is the problem of finding a minimum cograph completion of a graph obtained by adding one vertex to a cograph polynomial time solvable?*

To solve the first of these problems it might be enough to give a clever implementation of our naive extraction algorithm. For the second one, however, a clever implementation does not help as long as we output an explicit representation of the completed graph. All the other known algorithms that compute minimal completions in linear time [19, 22, 36], in fact, use some implicit representation. For cographs we can always use cotrees, but there are also other interesting representations that might be even more suitable for our problem, like the vertex ordering suggested in [9].

The third problem can be viewed as a generalization of the problem solved in [35], in which the addition of an edge instead of a vertex is considered.

Acknowledgement The authors would like to express their thanks to Pinar Heggernes for her helpful suggestions which improved the presentation of the paper.

References

[1] A. V. Aho, I. E. Hopcroft, and J. D. Ullman. The Design and Analysis of Computer Algorithms. *Addison-Wesley*, 1974. 3

[2] A. Berry, P. Heggernes, and G. Simonet. The minimum degree heuristic and the minimal triangulation process. In *Proceedings WG 2003 - 29th Workshop on Graph Theoretic Concepts in Computer Science*, pages 58–70, 2003. LNCS 2880. 1

[3] A. Berry, P. Heggernes, and Y. Villanger. A vertex incremental approach for dynamically maintaining chordal graphs. *Discrete Math.*, 306:318 – 336, 2006. 4

[4] J. Blair, P. Heggernes, and J. A. Telle. A practical algorithm for making filled graphs minimal. *Theoretical Computer Science*, 250:125–141, 2001. 1

[5] H. L. Bodlaender and A. M. C. A. Koster. Safe separators for treewidth. *Discrete Math.*, 306:337–350, 2006. 1

[6] H.L. Bodlaender, T. Kloks, D. Kratsch, and H. Müller. Treewidth and minimum fill-in on d-trapezoid graphs. *J. Graph Algorithms Appl.*, 2(5):1–28 ,1998. 1

[7] V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: grouping the minimal separators. *SIAM J. Comput.*, 31(1):212–232 2001. 1

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

[9] A. Bretscher, D. Corneil, M. Habib, and C. Paul. A Simple Linear Time LexBFS Cograph Recognition Algorithm. *SIAM J. Disc. Math.*, to appear. 5

[10] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99(1):367–400, 2000. 1

[11] P. Burzyn, F. Bonomo, and G. Durán. NP-completeness results for edge modification problems. *Disc. Appl. Math.*, 154:1824–1844, 2006. 1

[12] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. 1

[13] D.G. Corneil, H. Lerchs, and L.K. Stewart. Complement reducible graphs. *Disc. Appl. Math.*, 3:163 – 174, 1981. 1, 2.1

[14] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14:926 – 934, 1985. 1, 2.1, 3, 4.2

[15] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf power problems. *Algorithmica*, 44(4):363–381, 2006. 1

[16] F. V. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. In *Proceedings ICALP 2004*, pages 568–580, 2004. Springer LNCS 3142. 1

[17] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *J. Comput. Bio.*, 2(1):139–152, 1995. 1

[18] M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *J. Algorithms*, 19:449 – 473, 1995. 1

[19] P. Heggernes and F. Mancini. Minimal split completions of graphs. In *LATIN 2006: Theoretical Informatics*, pages 592–604. Springer Verlag, 2006. LNCS 3887. 1, 5

[20] P. Heggernes and F. Mancini. Dinamically maintaining split graphs. *Tech report:* http://www.ii.uib.no/∼federico/papers/dynsplit-rev2.pdf. 1

[21] P. Heggernes, F. Mancini, and C. Papadopoulos. Minimal comparability completions of arbitrary graphs. *Disc. Appl. Math.*, to appear. 1, 4, 5

[22] P. Heggernes and C. Papadopoulos. Single-edge monotonic sequences of graphs and linear-time algorithms for minimal completions and deletions. In *Proceedings of COCOON 2007 - 13th Annual International Conference on Computing and Combinatorics*, pages 406–416. Springer Verlag, 2007. LNCS 4598. 1, 3, 5

[23] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval Completion is Fixed Parameter Tractable. In *Proceedings of STOC 2007 - 39th Annual ACM Symposium on Theory of Computing*, pages 374–381, 2007. 1

[24] P. Heggernes, K. Suchan, I. Todinca, and Y. Villanger. Characterizing minimal interval completions: Towards better understanding of profile and pathwidth. In *Proceedings of STACS 2007 - 24th International Symposium on Theoretical Aspects of Computer Science*, pages 236 - 247. Springer Verlag, 2007. LNCS 4393. 1

[25] P. Heggernes, J. A. Telle, and Y. Villanger. Computing minimal triangulations in time $O(n^\alpha \log n) = o(n^{2.376})$. In *Proceedings of SODA 2005 - 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 907–916, 2005. 1

[26] H. Kaplan, R. Shamir, and R.E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum Fill-in and Physical Mapping. In *Proceedings of FOCS 2004 - 35th Annual Symposium on Foundations of Computer Science*, pages 780–791, 2004. 1

[27] T. Kashiwabara and T. Fujisawa. An NP-complete problem on interval graphs. *IEEE Symp. of Circuits and Systems*, pages 82–83, 1979. 1

[28] T. Kloks, D. Kratsch, and C. K. Wong. Minimum fill-in on circle and circular-arc graphs. *Journal of Algorithms*, 28(2):272–289, 1998. 1

[29] T. Kloks, D. Kratsch, and J. Spinrad. On treewidth and minimum fill-in of asteroidal triple-free graphs. *Theor. Comput. Sci.*, 175(2):309–335,1997. 1

[30] E. El-Mallah and C. Colbourn. The complexity of some edge deletion problems. *IEEE Transactions on Circuits and Systems*, 35:354 – 362, 1988. 1

[31] F. Mancini. Minimum fill-in and treewidth of split+ *ke* and split+*kv* graphs. In *Proceedings of ISAAC'07 - 18th International Symposium on Algorithms and Computation*, pages:881–892, 2007. LNCS 4835. 1

[32] D. Meister Computing treewidth and minimum fill-in for permutation graphs in linear time. In *Proceedings of WG 2005 - 31st International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 91–102, 2005. LNCS 3787. 1

[33] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Disc. Appl. Math.*, 113:109–128, 2001. 1

[34] A. Natanzon, R. Shamir, and R. Sharan. A polynomial approximation algorithm for the minimum fill-in problem. In *Proceedings of STOC'98 - 30th Annual ACM Symposium on Theory of Computing*, pages 41–47, 1998. 1

[35] S.D. Nikolopoulos and L. Palios. Adding an edge in a cograph. In *Graph Theoretic Concepts in Computer Science - WG 2005*, pages 214 – 226. Springer Verlag, 2005. LNCS 3787. 5

[36] I. Rapaport, K. Suchan, and I. Todinca. Minimal proper interval completions. In *Proceedings of WG 2006 - 32nd International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 217–228. Springer Verlag, 2006. LNCS 4271. 1, 5, 5

[37] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, New York, 1972. 1

[38] K. Suchan and I. Todinca. Minimal interval completion through graph exploration. In *Proceedings of ISAAC 2006 - 17th International International Symposium on Algorithms and Computation*, pages 517–526. Springer Verlag, 2006. LNCS 4288. 1

[39] Y. Villanger. Improved exponential-time algorithms for treewidth and minimum fill-in. In *Proceedings of LATIN 2006 - 7th Latin American Theoretical Informatics Symposium*, pages 800–811, 2006. LNCS 3887. 1

[40] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981. 1