Linear-time Constant-ratio Approximation Algorithm and Tight Bounds for the Contiguity of Cographs

Christophe Crespelle, Philippe Gambette

Linear-time Constant-ratio Approximation Algorithm and Tight Bounds for the Contiguity of Cographs

Christophe Crespelle[1] and Philippe Gambette[2]

[1] Université Claude Bernard Lyon 1, DNET/INRIA, LIP UMR CNRS 5668, ENS de Lyon, Université de Lyon, christophe.crespelle@inria.fr,
[2] Université Paris-Est, LIGM UMR CNRS 8049, Université Paris-Est Marne-la-Vallée, 5 boulevard Descartes, 77420 Champs-sur-Marne, France,
philippe.gambette@univ-mlv.fr

Abstract. In this paper we consider a graph parameter called *contiguity* which aims at encoding a graph by a linear ordering of its vertices. We prove that the contiguity of cographs is unbounded but is always dominated by $O(\log n)$, where n is the number of vertices of the graph. And we prove that this bound is tight in the sense that there exists a family of cographs on n vertices whose contiguity is $\Omega(\log n)$. In addition to these results on the worst-case contiguity of cographs, we design a linear-time constant-ratio approximation algorithm for computing the contiguity of an arbitrary cograph, which constitutes our main result. As a by-product of our proofs, we obtain a min-max theorem, which is worth of interest in itself, stating equality between the rank of a tree and the minimum height its path partitions.

Introduction

In many contexts, such as genomics, biology, physics, linguistics, computer science and transportation for examples, industrials and academics are led to algorithmically treat large dataset organised in the form of networks or graphs. The algorithms used to do so generally make extensive use of *neighborhood queries*, which, given a vertex x of a graph G, ask for the list of neighbors of x in G. Therefore, as pointed out by [1], due to the huge size of the graphs considered, finding compact representations of a graph providing optimal-time neighborhood queries is a crucial issue in practice.

One possible way to achieve this goal is to find an order σ on the vertices of G such that the neighborhood of each vertex x of G is an interval in σ. In this way, one can store the list of vertices of the graph in the order defined by σ and assign two pointers to each vertex: one toward its first neighbor in σ and one toward its last neighbor in σ. Therefore, one can answer adjacency queries on vertex x simply by listing the vertices appearing in σ between its first and last pointer. It must be clear that such an order on the vertices of G does not exist for all graphs G. Nevertheless, this idea turns out to be quite efficient in practice and some compression techniques are precisely based on it [2, 3]: they try to find orders of the vertices that group the neighborhoods together, as much as possible.

When one relaxes the constraints of the initial problem by rather asking for the minimum k such that there exists an order σ on the vertices of G where the neighborhood of each vertex is split in at most k intervals, this gives rise to a graph parameter called the *contiguity* of G [4]. This parameter was originally introduced in the broader context of binary matrices under the name k-consecutive-ones property [5]. It is worth to note that there are two variants of the parameter, respectively called *open contiguity* and *closed contiguity*, depending on whether one considers open neighborhoods (excluding the vertex itself) or closed neighborhoods (always containing the considered vertex). But this distinction is not fundamental as the two parameters always differ by at most one.

Here, we are interested in determining what is the worst-case contiguity for the cographs on n vertices, which are the graphs having no induced P_4 (path on 4 vertices). We also design an approximation algorithm that computes the contiguity of any cograph up to a constant ratio, in linear time with regard to the size of the input.

Related works. Only very little is known about the contiguity of graphs. Only the class of graphs having open contiguity 1 and the class of graphs having closed contiguity 1 have been characterized: the former are biconvex graphs [6] and the latter are proper interval graphs [7]. But the classes of graphs having contiguity at most k, where k is an integer greater than 1, have not been characterized, even for $k = 2$. Actually, closed contiguity has initially been studied in the context of $0-1$ matrices and [5, 8, 9] showed that deciding whether an arbitrary graph has closed contiguity at most k is NP-complete for any fixed $k \geq 2$. For arbitrary graphs again, [10] (Corollary 3.4) gave an upper bound on the value of closed contiguity which is $n/4 + O(\sqrt{n \log n})$, whose interest lies in the constant $1/4$ since it is clear that the contiguity of a graph is always less than $n/2$ (where n is the number of vertices of the graph). Finally, let us mention that [4] showed that the contiguity is unbounded for interval graphs as well as for permutation graphs, and that it can be up to $\Omega(\log n / \log \log n)$.

Our results. In this paper, we show that even for cographs, the contiguity is unbounded, but is dominated by $O(\log n)$ for a cograph on n vertices. To this purpose, we show (in Section 4) that the contiguity of a cograph G is mathematically equivalent the maximum height of a complete binary tree included (as a minor) in the cotree of G. This also allows us to exhibit a family of cographs $(G_n)_{n \in \mathbb{N}}$ on n vertices whose asymptotic contiguity is $\Omega(\log n)$, which implies that our $O(\log n)$ bound is tight. Furthermore, from this result, we derive in Section 5 a constant-ratio approximation algorithm that computes the contiguity of an arbitrary cograph (up to a multiplicative constant) in linear time wrt. the size of the input, that is $O(n)$ time provided that the input cograph is given by its cotree (see Section 1 for a definition). In addition, our algorithm can also provide a linear ordering σ of the vertices of G, together with the pointers from each vertex to the at most k intervals partitioning its neighborhood, that realizes a k which is in a constant ratio from the optimal one, i.e. the contiguity of G. In this case, the complexity of our algorithm is linear wrt. the size of the output, that is $O(kn)$ time.

As a by-product of our proofs, we also establish in Section 2 a min-max theorem which is worth of interest in itself: the maximum height of a complete binary tree included (as a minor) in a tree T (known as the *rank* of tree T [11, 12]) is equal to the minimum height of a partition of T into vertex-disjoint paths.

1 Preliminaries.

All graphs considered here are finite, undirected, simple and loopless. In the following, G denotes a graph, V (or $V(G)$ to avoid ambiguity) denotes its vertex set and E (or $E(G)$) its edge set. We use the notation $G = (V, E)$ and we denote $|V| = n$. The set of subsets of V is denoted by 2^V. An edge between vertices x and y will be arbitrarily denoted by xy or yx. The (open) neighborhood of x is denoted by $N(x)$ (or $N_G(x)$ to avoid ambiguity) and its closed neighborhood by $N[x] = N(x) \cup \{x\}$. The subgraph of G induced by the subset of vertices $X \subseteq V$ is denoted by $G[X] = (X, \{xy \in E \mid x, y \in X\})$.

Cographs are the graphs that do not have any P_4 (path on 4 vertices) as induced subgraph. They are also known to be the graphs G admitting a *cotree*, i.e. a rooted tree T whose leaves are the vertices of G, and whose internal nodes are labelled *series* or *parallel* with the following property: any two vertices x and y of G are adjacent iff the least common ancestor u of leaves x and y in T is a series node. Otherwise, if u is a parallel node, x and y are not adjacent.

For a rooted tree T and a node $u \in T$, the depth of u in T is the number of edges in the path from the root to u (the root has depth 0). The *height* of T, denoted by *height*(T) or simply $h(T)$, is the greatest depth of its leaves. For a rooted tree T, the subtree of T rooted at u, denoted by T_u, is the tree induced by node u and all its descendants in T. In the following, the notion of *minors* of rooted trees is central. This is a special case of minors of graphs (see e.g. [13]), for which we give a simplified definition in the context of rooted trees.

Definition 1.1. *The* contraction of edge *uv in a rooted tree T, where u is the parent of v, consists in removing v from T and assigning its children (if any) to node u.*
A rooted tree T' is a minor *of a rooted tree T if it can be obtained from T by a sequence of edge contractions.*

Let us now formally define the contiguity of a graph.

Definition 1.2. *A* closed p-interval-model *(resp. open p-interval-model) of a graph $G = (V, E)$ is a linear order σ on V such that $\forall v \in V, \exists (I_1, \ldots, I_p) \in (2^V)^p$ such that $\forall i \in [\![1, p]\!]$, I_i is an interval of σ and $N[v] = \bigcup_{1 \leq i \leq p} I_i$ (resp. $N(v) = \bigcup_{1 \leq i \leq p} I_i$).*
The closed contiguity *(resp.* open contiguity*) of G, denoted by $cc(G)$ (resp. $oc(G)$) , is the minimum integer p such that there exists a closed p-interval-model (resp. open p-interval-model) of G.*

It is worth to note that the closed and open contiguity never differ by more than one. Indeed, given a closed k-interval model of a graph G, we directly get an open $k+1$-interval model for G by simply splitting, for each vertex x, the interval containing x into (at most) two intervals. Conversely, adding (at most) one trivial interval $\{x\}$ for each vertex x in an open k-interval-model results in a closed $(k+1)$-interval model. Therefore, in all the rest of the paper, we only consider closed contiguity, but all our results also hold for open contiguity. In the following, we abusively extend the notion of contiguity to cotrees referring to the contiguity of their associated cograph.

2 Some general results on the rank of trees

In this section, we give two general results on trees which will play a key role in the rest of the paper. The first result links the rank of a tree T with the minimum height of a partition of vertices of T into paths.

The *rank* [11, 12] of a tree T is the maximal height of a complete binary tree obtained from T by edge contractions, that is: $rank(T) = \max\{h(T') \mid T' \text{ complete binary tree, minor of } T\}$. A *path partition* of a tree T is a partition $\{P_1, \ldots, P_k\}$ of $V(T)$ such that for any i, the subgraph $T[P_i]$ of T induced by P_i is a path, as shown in Figure 1(a). The *partition tree* of a path partition $\mathcal{P}$, denoted by $T_p(\mathcal{P})$, is the tree whose nodes are P_i's and where the node of $T_p(\mathcal{P})$ corresponding to P_i is the parent of the node corresponding to P_j iff some node of P_i is the parent in T of the root of P_j (see Fig. 1(b)). The height of a path partition $\mathcal{P}$ of a tree T, denoted by $h(\mathcal{P})$, is the height $h(T_p(\mathcal{P}))$ of its partition tree. The *path-height* of T is the minimal height of a path partition of T, that is $ph(T) = \min\{h(\mathcal{P}) \mid \mathcal{P} \text{ path partition of } T\}$.

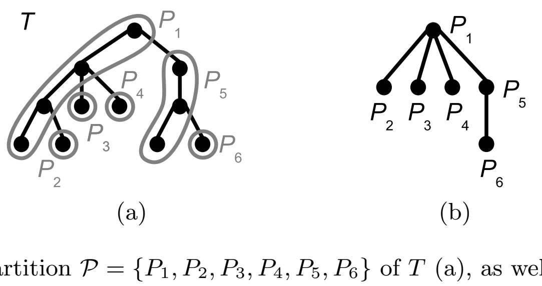

Fig. 1. A tree T and a path partition $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ of T (a), as well as the partition tree of $\mathcal{P}$ (b).

Theorem 2.1. *For any rooted tree T, $rank(T) = ph(T)$.*

Sketch of proof. It is not difficult to show that the path-height of a tree T is at least the path-height of any tree T' included in T as a minor. On the other hand, a simple recursion on the height of a complete binary tree shows that its path-height is at least its height. It follows that the path-height of any tree T is at least the maximum height of a complete binary tree T' included as a minor in T, that is $ph(T) \geq rank(T)$. The converse inequality, namely $ph(T) \leq rank(T)$, can be shown by induction on $rank(T)$. Indeed, consider a tree T such that $rank(T) = k+1$. The nodes u of T such that $rank(T_u) = k+1$ form a path P containing the root of T. By definition, any node $v \notin P$ is such that $rank(T_v) \leq k$. Then, by the induction hypothesis, $ph(T_v) \leq rank(T_v) \leq k$. And it follows that $ph(T) \leq max_{v \notin P} ph(T_v) + 1 \leq k + 1 = rank(T)$. A detailed version of this proof is given in Appendix A, in Theorem 5.1. □

We now consider *bicolored trees*, i.e. trees whose nodes are colored either black or white. We define the *black rank* (resp. *white rank*), denoted $r_B(T)$ (resp. $r_W(t)$), of a bicolored tree T as the maximum height of an entirely black (resp. entirely white) complete binary tree being a minor of T.

Theorem 2.2. *For any bicolored complete binary tree T, $r_B(T) + r_W(T) \geq h(T) - 1$.*

Sketch of proof. The proof is by induction on $h(T)$. Consider a complete binary bicolored tree T of height $k+1$, whose root is colored black wlog. and has two children denoted by u_1 and u_2. If $r_B(T_{u_1}) = r_B(T_{u_2})$, then $r_B(T) = r_B(T_{u_1}) + 1$. And since $r_W(T) \geq r_W(T_{u_1})$, by the induction hypothesis, we obtain $r_B(T) + r_W(T) \geq h(T_{u_1}) + 1 - 1 = h(T) - 1$. On the other hand, if $r_B(T_{u_1}) \neq r_B(T_{u_2})$, then assume wlog. that $r_B(T_{u_1}) > r_B(T_{u_2})$. Then, we have $r_B(T) + r_W(T) \geq r_B(T_{u_1}) + r_W(T_{u_2}) \geq r_B(T_{u_2}) + 1 + r_W(T_{u_2})$, and by the induction hypothesis, we obtain the desired inequality for T. A detailed version of this proof is given in Appendix A, in Theorem 5.2. □

3 An upper bound for the contiguity of cographs

We now prove that the contiguity of any cograph is linearly bounded by the rank of its cotree T (Theorem 3.1 below) by using a path partition of T of minimal height h. Our proof is by induction on h and Lemma 3.1 below constitutes the recursive encoding step of our proof.

In a path partition of T, the path containing the root naturally induces a partition of the leaves of T, i.e. the vertices of the corresponding cograph G, as described in the following definition. A *root-path decomposition* (see Fig. 2) of a rooted tree T is a set $\{T_1, \ldots, T_p\}$ of disjoint subtrees of T, with $p \geq 2$, such that every leaf of T belongs to some T_i, with $i \in [1..p]$, and the sets of parents in T of the roots of T_i's is a path containing the root of T.

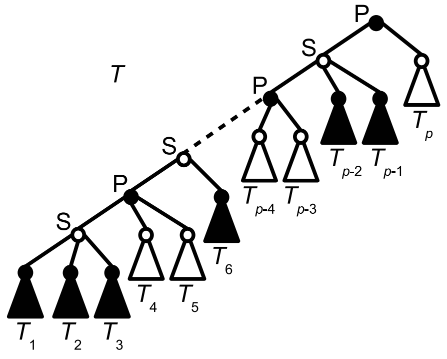

Fig. 2. The root-path decomposition $\{T_1, \ldots, T_p\}$ of a rooted tree T.

Lemma 3.1 (Caterpillar Composition Lemma). *Given a cograph $G = (V, E)$ and a root-path decomposition $\{T_i\}_{1\leq i\leq p}$ of its cotree, where X_i is the set of leaves of T_i, $cc(G) \leq 2 + \max_{i\in[1..p]} cc(G[X_i])$.*

Sketch of proof. It is straightforward to check that for any $i \in [1..p]$ and for any vertex $x \in X_i$, the neighbors of x that are not in X_i are split in at most two intervals in the order σ given on Fig. 3. Therefore, by choosing for σ an order on each X_i that realizes the contiguity of $G[X_i]$, we obtain that in σ, the neighborhood of any vertex $x \in X_i$ of G is split in at most $2 + cc(G[X_i])$ intervals, which proves the lemma. A detailed version of this proof is given in Appendix B, in Lemma 5.3. □

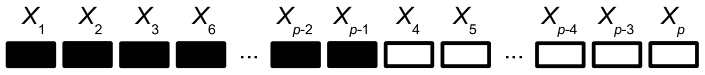
X_1 X_2 X_3 X_6 ... X_{p-2} X_{p-1} X_4 X_5 ... X_{p-4} X_{p-3} X_p

Fig. 3. The general structure of the order σ used in Lemma 3.1 for the root-path decomposition of Fig 2.

From this, we deduce an upper bound on the contiguity of a cograph depending on the rank of its cotree.

Theorem 3.1. *For any cograph G having cotree T, $cc(G) \leq 2\,rank(T) + 1$.*

Sketch of proof. The proof is by induction on $rank(T)$. The inequality holds for $rank(T) = 0$. Consider a cograph G whose cotree T is of rank $k+1$, with $k \geq 0$. The nodes u of T such that $rank(T_u) = k + 1$ form a path containing only internal nodes of T and containing its root. Therefore, this path induces a root-path decomposition $\{T_1, \ldots, T_p\}$ of T as shown on Fig. 2. By definition, this root-path decomposition is such that for any $i \in [1..p]$, tree T_i is of rank at most k. Then, by the induction hypothesis, $cc(G[X_i]) \leq 2\,rank(T_i)+1$. From Lemma 3.1, we have that $cc(G) \leq 2+\max_{i\in[1..p]} cc(G[X_i])$. It follows that $cc(G) \leq 2 + \max_{i\in[1..p]}\{2\,rank(T_i) + 1\} \leq 2 + 2k + 1 = 2(k + 1) + 1$, which ends the induction and the proof of Theorem 3.1. A detailed version of this proof is given in Appendix B, in Theorem 5.3. □

As the rank of a tree T is bounded by the logarithm of its number of leaves, we directly obtain the following corollary.

Theorem 3.2. *If $G = (V, E)$ is a cograph, then $cc(G) \leq 2\log_2 |V| + 1$.*

4 A lower bound for the contiguity of cographs

In this section, we show that the rank of a cotree also provides a lower bound on the contiguity of its associated cograph. Together with the result of the previous section, this give that the contiguity of a cograph and the rank of its cotree are mathematicaly equivalent functions, which is at the core of the approximation algorithm we present in next section. We also use the lower bound to exhibit a family of cographs on n vertices whose contiguity is at least $\Omega(\log n)$, showing that the upper bound of the previous section is tight.

Lemma 4.1. *Let G_k be the underlying undirected graph of the transitive closure of the directed rooted complete ternary tree T^k of depth $k \geq 0$. Then, we have $cc(G_k) \geq k + 1$.*

Proof: We prove it by induction on k. We obviously have $cc(G_0) \geq 1$.

Now, suppose that $cc(G_k) \geq k+1$ for some $k \geq 0$, and let us show that $cc(G_{k+1}) \geq k+2$. Consider a $cc(G_{k+1})$-interval-model of G_{k+1}, and denote by σ the corresponding order on the vertices of G.

We denote by r the root of T^{k+1}, and by v_1, v_2 and v_3 its three children in T^{k+1}. Since there are at most two vertices of T^{k+1} that are next to r in σ, then there exists $i \in [\![1, 3]\!]$ such that no vertex of $T^{k+1}_{v_i}$ is next to r in σ. Denoting by G_{v_i} the subgraph of G_{k+1} induced by the vertices of $T^{k+1}_{v_i}$, the induction hypothesis gives that $cc(G_{v_i}) \geq k + 1$. In particular, this implies that in the restriction of σ to the vertices of G_{v_i}, there exists some $v \in G_{v_i}$ such that its neighborhood in G_{v_i} is split into at least $k + 1$ intervals. Since r is adjacent to v in G_{k+1}, and since the at most two vertices next to r in σ are not in the neighborhood of v, it follows that the neighborhood of v requires one more interval for r in σ. Thus, $cc(G_{k+1}) \geq k + 2$. □

It is worth to note that G_k is a cograph. Indeed, its cotree can be recursively built as follows. The cotree of G_1 (which is also known as the *claw* graph) is made of one series root having two children: a leaf and a parallel node having three leaf children. And replacing these three leaves by three copies of the cotree of G_k, for any $k \geq 1$, results in the cotree of G_{k+1}. From now on, in order to use the result of Theorem 2.2, we considered cotrees bicolored as explained in the following definition. The *bicolored cotree* of a cograph G is its cotree where the parallel nodes are colored black and the series nodes are colored white.

Lemma 4.2. *For any cograph G, whose bicolored cotree has a white root and black rank at least $2k$, we have $cc(G) \geq k + 1$.*

Sketch of proof. We prove by induction on k that G contains G_k as an induced subgraph, which has contiguity at least $k+1$ from Lemma 4.1. Since G has black rank at least $2k$, its cotree T contains, as a minor, an entirely black complete binary tree T'. Consider the two nodes at depth 1 in T', they have at least four different children in T. For three of them u_1, u_2 and u_3, take one leaf of their subcotree in T: since the nodes of T' are all black (i.e. parallel), this gives three independent vertices denoted by x_1, x_2 and x_3. Now recall that the root of T is white (i.e. series) and must have a child which is not an ancestor of the root of T'. Take a leaf y being a descendant of this child: y, x_1, x_2 and x_3 induce the claw graph, i.e. the graph G_1 of Lemma 4.1.

Finally, note that for any $u \in \{u_1, u_2, u_3\}$, u is necessarily white (as a parallel node of a cotree has only series, or leaf, children) and the cotree T_u has black rank at least $2k - 2$. Then, from the induction hypothesis, the cograph associated to T_u contains G_{k-1} as an induced subgraph. And from the recursive construction of the cotree of G_k that precedes Lemma 4.2, we conclude that G contains G_k as an induced subgraph.

A detailed version of this proof is given in Appendix C, in Lemma 5.5. □

Theorem 4.1. *For any cograph G and its cotree T, $cc(G) \geq (rank(T) - 7)/4$.*

Proof: Consider the bicolored cotree T of G. Then, from Lemma 2.2, $r_B(T) + r_W(T) \geq rank(T) - 1$, so either $r_B(T) \geq (rank(T) - 1)/2$, or $r_W(T) \geq (rank(T) - 1)/2$.

In the first case, $r_B(T) \geq 2((rank(T) - 3)/4) + 1$, so T has a black complete binary tree of height $2((rank(T) - 3)/4)$, below a white node, as a minor, so Lemma 4.2 implies that $cc(G) \geq (rank(T) - 3)/4 + 1 \geq (rank(T) - 7)/4$.

In the second case, the bicolored cotree $T_{\bar{G}}$ of the complement of G is simply the bicolored cotree of G where series and parallel nodes, and so black and white nodes, have been exchanged. Then, we have $r_B(T_{\bar{G}}) = r_W(T) \geq (rank(T) - 1)/2$. Lemma 4.2 implies similarly as above that $cc(\bar{G}) \geq (rank(T) - 3)/4 + 1$. We can easily check that $cc(\bar{G}) \leq cc(G) + 2$ (the closed neighborhood of any vertex x in $\bar{G}$ is composed by at most $cc(G) + 1$ intervals surrounding the intervals of the closed neighborhood of x in G, plus possibly one interval for vertex x), so $cc(G) \geq (rank(T) - 3)/4 + 1 - 2 = (rank(T) - 7)/4$.
□

From Theorem 4.1 we obtain that the contiguity of cographs is unbounded. Moreover, since the rank of a complete binary tree is, by definition, its height, it follows that the family of cographs having a complete binary cotree reaches the $O(\log n)$ upper bound of Theorem 3.2, showing that this bound is tight.

Corollary 4.1. *For any cograph G whose cotree is a complete binary tree, $cc(G) = \Omega(\log n)$.*

As a direct consequence of Theorems 4.1 and 3.1, we obtain the equivalence between the contiguity of a cograph and the rank of its cotree, which is at the core of the approximation algorithm presented in next section.

Corollary 4.2. *For any cograph G with cotree T, $cc(G) = \Omega(rank(T))$.*

5 An approximation algorithm for the contiguity of cographs

We now provide an algorithm that outputs an integer k and a k-interval-model of the cograph G given as input, such that k is in a constant ratio from the contiguity of G. Our algorithm takes as input a cograph G given by its cotree, which takes $O(n)$ space, and outputs a k-interval model of G in $O(kn)$ time, i.e. linear wrt. the size of the output. Alternatively, if only the value of k is needed, the algorithm runs in $O(n)$ time, which is linear wrt. the size of the input.

First step: approximation of the contiguity of G. The first step of our algorithm computes an integer k which is in a constant ratio from the contiguity of G. To this purpose, we simply compute recursively the rank of T_u for any node $u \in T$ by a bottom-up process: for a node u we have either i) $rank(T_u) = max_{v \text{ child of } u} rank(T_v)$ if this maximum is reached by only one child of u, or ii) $rank(T_u) = 1 + max_{v \text{ child of } u} rank(T_v)$ otherwise. Then, our algorithm outputs the value $k = 2\,rank(T_r) + 1$, where r is the root of T, and so $rank(T_r) = rank(T)$. This clearly takes $O(n)$ time.

Furthermore, the approximation ratio ρ of this algorithm is constant. Using Theorem 4.1 and the fact that $cc(G) \geq 1$, we deduce that $\rho = (2\,rank(T)+1)/cc(G) \leq (2\,rank(T)+1)/max(1, (rank(T)-7)/4)$. This function reaches its maximum of 23 for $rank(T) = 11$, and then the algorithm performs a 23-approximation of the closed contiguity (which can indeed be lower to $8 + \epsilon$, where ϵ is a positive constant as small as required, by using a constant-time precomputing step).

Second Step: building a k-interval model of G. We follow the construction provided by Theorem 3.1 and Lemma 3.1 (see Fig. 3) in order to output a k-interval-model of G, where $k = 2\,rank(T)+1$. During this process, we build the order σ on the vertices as well as a table $Neighborhoods$ of n tables of $2k$ pointers to the bounds of the intervals of each vertex in order σ. To this purpose, we call the recursively defined routine $Build(u)$, where u is a node of T, on the root r of T. $Build(u)$ outputs an order σ_u on the subset X_u of vertices of G being the leaves of T_u and table $Neighborhoods_u$ containing the pointers of the vertices of X_u toward σ_u. Routine $Build(u)$ proceeds in three steps as follows.
i) thanks to the ranks computed in the first step of the algorithm, $Build(u)$ finds the subset P_u of nodes $\tilde{u}$ of T_u such that $rank(T_{\tilde{u}}) = rank(T_u)$, and the subset C_u of children in T of nodes $\tilde{u} \in P_u$.
ii) $Build(u)$ recursively calls $Build(v)$ for all nodes $v \in C_u$.
iii) $Build(u)$ builds σ_u by concatenating all the orders σ_v returned by the recursive calls as shown on Fig. 3, and builds $Neighborhoods_u$ by merging all the tables $Neighborhoods_v$ returned by the recursive calls. Then, for each $v \in C_u$ and for each vertex $x \in X_v$, we add to $Neighborhoods_u$ the pointers of x toward the at most two intervals of σ_u formed by its neighbors that are not in X_v, as explained in the proof of Lemma 3.1.

The terminal case of Routine $Build(u)$ is when u is a leaf of T, for which the computation of σ_u and $Neighborhoods_u$ is trivial and takes constant time. The fact that a call to $Build(r)$ indeed gives the desired k-interval model of G comes from the fact that the routine follows the constructive proof of Theorem 3.1. Let us analyze its complexity. Step **i)**, Step **ii)** and the construction of σ_u in Step **iii)** take $O(|C_u| + |P_u|)$, that is $O(n)$ for the whole process on tree T. In Step **iii)**, the merge of table $Neighborhoods$ and the addition of the pointers to the two new intervals of each vertex take $O(X_u)$ time. It turns out that, during the whole process on T, a vertex x will be involved in at most h different sets X_u, where h is the height of the path partition of T defined by the set of paths P_u computed along the process. From Theorem 3.1, $h = rank(T) = (k-1)/2$. Thus, the total computation time of the k-interval model output by our algorithm is $O(kn)$ time.

Conclusion

We showed that the contiguity of a cograph is equivalent to the maximum height of a complete binary tree contained in its cotree as a minor. From this, we obtained a tight $O(\log n)$ upper bound on the maximum contiguity of a cograph on n vertices. Even more interesting, this allowed us to design a linear time algorithm that does not only compute an approximation of the contiguity of a cograph G but also provides a k-interval model realizing a k which is in a constant-ratio to the optimal one, i.e. the contiguity of G. Then, the first question raised by our work is whether it is possible to compute efficiently the exact value of the contiguity of a cograph and to provide a model realizing this optimal value. Another key perspective is to extend our results to larger classes of graphs, such as permutation graphs (which are a proper generalization of cographs) and interval graphs. Does the $O(\log n)$ upper bound still hold for those graphs? Is it possible to compute efficiently an exact or approximated value of their contiguity?

Acknowledgments. The authors thank Pierre Charbit and Stéphan Thomassé for useful discussions on the subject, as well as George Oreste Manoussakis for proofreading a draft of this article. This work was partially supported by the PEPS-C1P CNRS project.

References

1. G. Turan, On the succinct representation of graphs, Discr. Appl. Math. 8 (1984) 289–294.
2. P. Boldi, S. Vigna, The webgraph framework I: compression techniques, in: WWW'04, ACM, 2004, pp. 595–602.
3. P. Boldi, S. Vigna, Codes for the world wide web, Internet Mathematics 2 (4) (2005) 407–429.
4. C. Crespelle, P. Gambette, Efficient neighbourhood encoding for interval graphs and permutation graphs and $O(n)$ breadth-first search, in: 20^{th} International Workshop on Combinatorial Algorithms (IWOCA'09), no. 5874 in LNCS, 2009, pp. 146–157.
5. P. Goldberg, M. Golumbic, H. Kaplan, R. Shamir, Four strikes against physical mapping of DNA, Journal of Computational Biology 2 (1) (1995) 139–152.
6. A. Brandstädt, V. Le, J. Spinrad, Graph Classes: a Survey, SIAM Monographs on Discrete Mathematics and Applications, 1999.
7. F. Roberts, Representations of indifference relations, Ph.D. thesis, Stanford University (1968).
8. D. Johnson, S. Krishnan, J. Chhugani, S. Kumar, S. Venkatasubramanian, Compressing large boolean matrices using reordering techniques, in: Proceedings of the Thirtieth international conference on Very Large Data Bases (VLDB'04), Vol. 30, 2004, pp. 13–23.
9. R. Wang, F. Lau, Y. Zhao, Hamiltonicity of regular graphs and blocks of consecutive ones in symmetric matrices, Discr. Appl. Math. 155 (17) (2007) 2312–2320.
10. C. Gavoille, D. Peleg, The compactness of interval routing, SIAM Journal on Discrete Mathematics 12 (4) (1999) 459–473.
11. D. H. A. Ehrenfeucht, Learning decision trees from random examples, Information and Computation 82 (3) (1989) 231–246.
12. R. Gavaldà, D. Thérien, Algebraic characterizations of small classes of boolean functions, in: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03), Vol. 2607 of LNCS, 2003, pp. 331–342.
13. L. Lováz, Graph minor theory, Bulletin of the American Mathematical Society 43 (1) (2006) 75–86.

Appendix

Appendix A) Some useful results on trees

We formally state and prove the claim in the end of Section 1 that “the closed and open contiguity never differ by more than one”.

Lemma 5.1. *For any graph G, $cc(G) \leq oc(G) + 1$ and $oc(G) \leq cc(G) + 1$.*

Proof: If we have an open k-interval-model for G based on order σ, we can make a closed k-interval-model of G by simply adding one interval for vertex x, when we are encoding the closed neighborhood of x by a union of intervals, so $cc(G) \leq oc(G) + 1$.

For the second inequality, if we have a closed k-interval-model, removing vertex x from the union of intervals encoding its neighborhood will, in the worst case, break one interval (the one containing x) into two, and therefore it may cost up to one more interval. So $oc(G) \leq cc(G) + 1$ □

We now give the details of the proof of Theorem 2.1. To this purpose, we first prove the result on rooted complete binary trees.

Lemma 5.2. *For any rooted complete binary tree T, $rank(T) = ph(T) = h(T)$.*

Proof: The fact that $h(T) = rank(T)$ is obvious from the definition of $rank(T)$. On the other hand, one can build a path partition $\mathcal{P}$ of height $h(T)$ by taking $\mathcal{P} = \{\{x\}\}_{x \in V(T)}$. In other words, each path in $\mathcal{P}$ is reduced to one single vertex of T. In this way, $T_p(\mathcal{P}) = T$ wich gives $ph(T) \leq h(T)$.

Conversely, we prove that $ph(T) \geq h(T)$ by induction on $h(T)$. This property is true for the trivial tree T with only one vertex, for which we have $ph(T) = 0 \geq 0 = h(T)$. Now, assume that it is true for any rooted complete binary tree of height $k \geq 0$, and consider a rooted complete binary tree T of height $k + 1$. Let $\mathcal{P}$ be a path partition of T such that $h(\mathcal{P}) = ph(T)$, and let P be the path of $\mathcal{P}$ containing the root r of T, as shown in Figure 4.

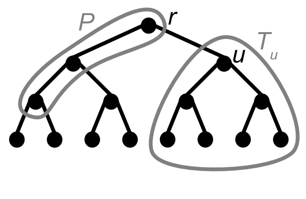

Fig. 4. The induction step of the proof that the path-height of a complete binary tree is at least as big as its height, in the proof of Lemma 5.2.

The path P does not contain at least one child u of r. Then, the subset $\mathcal{P}'$ of $\mathcal{P}$ made of those paths containing some node of T_u is a path partition of T_u. And from the induction hypothesis, we get $h(\mathcal{P}') \geq ph(T_u) \geq h(T_u) \geq k$. Since $h(\mathcal{P}) \geq h(\mathcal{P}') + 1$, we obtain $h(\mathcal{P}) \geq k + 1$, and so $ph(T) \geq k + 1$. This ends the induction and the proof of Lemma 5.2. □

Theorem 5.1. *For any rooted tree T, $rank(T) = ph(T)$.*

Proof: $\leq$: Let T' be a complete binary tree which is a minor of T of maximum height among such complete binary trees. By definition, we have $rank(T) = h(T')$ and from Lemma 5.2 we know that $h(T') = ph(T')$. It follows that $rank(T) = ph(T')$. Then, in order to prove $rank(T) \leq ph(T)$, we show that $ph(T') \leq ph(T)$.

Indeed, consider an arbitrary path partition $\mathcal{P}$ of T. We will build a path partition $\mathcal{P}'$ of T' whose height is at most the height of $\mathcal{P}$. By definition, tree T' can be obtained from T by a series of edge contractions. The partition $\mathcal{P}'$ we build is induced from $\mathcal{P}$ in the sense that we obtain it by updating partition $\mathcal{P}$ step by step along the series of edge contractions of T resulting in T'. Along this process, when an edge uv of the current tree T_{cur}, where u is the parent of v, is contracted in order to obtain the next tree T_{next}, we delete node v and assign its children to u, as in Definition 1.1 and as shown in Figure 5.

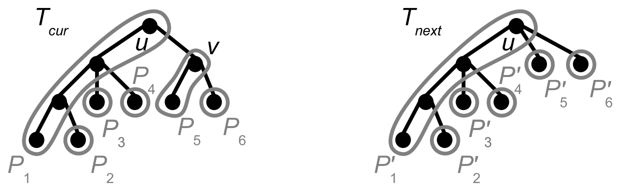

Fig. 5. Left: a tree T_{cur} with its path partition $\mathcal{P}_{cur} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ whose partition tree T_p^{cur} has height 2. Right: the tree T_{next} resulting from the contraction of edge uv in T_{cur}, together with its path partition $\mathcal{P}_{next} = \{P'_1, P'_2, P'_3, P'_4, P'_5, P'_6\}$. All the paths in $\mathcal{P}_{next}$ are the same as those in $\mathcal{P}_{cur}$, except P'_5 which has lost node v. The height of T_p^{next} decreased, it is now 1.

This ensures that any set P of nodes inducing a path in T_{cur} also induces a path in T_{next} (except in the case where $P = \{v\}$: P becomes the empty set after the contraction and disappears from the path partition we maintain). Consequently, a path partition $\mathcal{P}_{next}$ of T_{next} can be obtained from the partition $\mathcal{P}_{cur}$ of T_{cur} in the following way: for all the nodes remaining in T_{next} (i.e. all the nodes of T_{cur} except node v), we keep them in the same path of the partition as they were previously belonging to in $\mathcal{P}_{cur}$.

It is not difficult to see that the partition tree T_p^{next} of $\mathcal{P}_{next}$ defined this way has height at most that of T_p^{cur}. Indeed, from the definition of the partition tree (see the beginning of Section 2), during the contraction of edge uv, the only paths of $\mathcal{P}_{cur}$ that are likely to change their parent in T_p^{next} are those paths whose root is one of the children of v, as the other nodes of T_{cur} do not change their parent. For such a path $P_{v'}$ rooted at a child v' of v, the parent of $P_{v'}$ in T_p^{next} will be the path P_u containing u, which may be either its parent or its grandparent in T_p^{cur}, depending on whether nodes u and v are in the same path of $\mathcal{P}_{cur}$ or not. Since the only nodes of T_p^{cur} that change their parents in T_p^{next} change it for their grandparent in T_p^{cur}, it follows that $h(\mathcal{P}_{next}) \leq h(\mathcal{P}_{cur})$.

Consequently, at the end of the series of edge contractions that results in T', we obtain a path partition $\mathcal{P}'$ of T' such that $h(\mathcal{P}') \leq h(\mathcal{P})$. As this holds for any path partition $\mathcal{P}$ of T, we conclude that $ph(T') \leq ph(T)$.

$\geq$: We show that $rank(T) \geq ph(T)$ by induction on $rank(T)$. We use the following induction hypothesis $H(k)$: "for any tree T such that $rank(T) = k$, we have $ph(T) \leq rank(T)$".

Clearly, if $rank(T) = 0$, then T is a rooted path. Thus, there is a trivial partition of T into a single path and the path-height of this partition is $0 : ph(T) \leq rank(T)$.

Now, let $k \geq 0$ such that $\forall i \in [\![0, k]\!], H(k)$ holds. We show that $H(k+1)$ is satisfied. Let T be a tree such that $rank(T) = k+1$, and let S_{k+1} be the subset of nodes u of T such that $rank(T_u) = k+1$. We first show that S_{k+1} is a path of T containing the root.

Clearly, by definition of $rank(T)$, S_{k+1} contains the root of T, so let us show that it induces a path of T. First, note that if u is such that $rank(T_u) = k+1$ then for all ancestors v of u, $rank(T_v) = k+1$. Indeed, it is clear that $rank(T_v)$ is not less than $rank(T_u)$, and since $rank(T) = k+1$, we also have $rank(T_v) \leq k+1$. Now, assume for contradiction that there exist two nodes $u, v \in S_{k+1}$ such that none of them is the ancestor of the other. Let w be the least common ancestor of u and v in T. Let u' and v' be the two distinct children of w that are the ancestors of respectively u and v. From above, u' and v' are such that $rank(T_{u'}) = rank(T_{v'}) = k+1$. It follows that $rank(T_w) \geq k+2$: contradiction. Thus, all the nodes of S_{k+1} are comparable for the ancestor relationship, and since we already showed that all the ancestors of a node in S_{k+1} are in S_{k+1}, it follows that S_{k+1} induces a path of T containing the root.

Now, let $S' = \{u \in V(T) \setminus S_{k+1} \mid parent(u) \in S_{k+1}\}$, and let $u \in S'$. Clearly, since $u \notin S_{k+1}$, $rank(T_u) \leq k$ and the induction hypothesis implies that $ph(T_u) \leq k$. For any $u \in S'$, we denote by $\mathcal{P}_u$ a path partition of T_u of height at most k. Then, $\{S_{k+1}\} \cup \bigcup_{u \in S'} \mathcal{P}_u$ is a path partition of T and has height at most $k+1$. Thus, $ph(T) \leq k+1 = rank(T)$ and $H(k+1)$ holds, which ends the induction, showing that for any tree T, $rank(T) \geq ph(T)$. □

Theorem 5.2. *For any bicolored complete binary tree T, $r_B(T) + r_W(T) \geq h(T) - 1$.*

Proof: We prove it by induction on $h(T)$. This clearly holds for $h = 1$.

Suppose the property holds for some $h \geq 1$, and let T be a complete binary tree of height $h+1$. We show that the properties also hold for T. T is made of a root r with two children subtrees T_a and T_b. Let us assume wlog that the root of T is colored black.

If $r_B(T_a) = r_B(T_b)$, then $r_B(T) = r_B(T_a) + 1$. And since $r_W(T) \geq r_W(T_a)$ and $r_B(T_a) + r_W(T_a) \geq h(T_a) - 1$, then we have $r_B(T) + r_W(T) \geq h(T_a) + 1 - 1 = h(T) - 1$, and the property is satisfied.

On the other hand, if $r_B(T_a) \neq r_B(T_b)$, let us assume wlog. that $r_B(T_a) > r_B(T_b)$. We clearly have $r_B(T) \geq max(r_B(T_a), r_B(T_b))$ as well as $r_W(T) \geq max(r_W(T_a), r_W(T_b))$. Since $r_B(T_a) > r_B(T_b)$, $max(r_B(T_a), r_B(T_b)) \geq r_B(T_b) + 1$. On the other hand, $max(r_W(T_a), r_W(T_b)) \geq r_W(T_b)$, so $r_B(T) + r_W(T) \geq max(r_B(T_a), r_B(T_b)) + max(r_W(T_a), r_W(T_b)) \geq r_B(T_b) + 1 + r_W(T_b) \geq h(T_b) - 1 + 1$ (from the induction hypothesis on T_b) and $h(T_b) = h(T) - 1$, so finally $r_B(T) + r_W(T) \geq h(T) - 1$. □

Appendix B) An upper bound for the contiguity of cographs

Lemma 5.3 (Caterpillar Composition Lemma). *Given a cograph $G = (V, E)$ and a root-path decomposition $\{T_i\}_{1 \leq i \leq p}$ of its cotree,*

$$cc(G) \leq 2 + \max_{i \in [\![1,p]\!]} cc(G[X_i]),$$

where X_i is the set of leaves of T_i.

Proof: For convenience of description of the closed contiguity model of G, we assume wlog. that the T_i's are numbered in such a way that for any $i, j \in [\![1, p]\!]$, if the parent of the root of T_i is a strict ancestor of the parent of the root of T_j then $i > j$. Moreover, we color the trees T_i of the root-path decomposition in the following way: if the root of T_i is a series node, T_i is colored white, otherwise T_i is colored black.

Let σ_i be a closed contiguity model realizing the closed contiguity of $G[X_i]$, where X_i is the set of leaves of T_i. We now build an order σ on the vertices of G such that for each $x \in V(G)$, $N[x]$ is the union of at most $l+2$ intervals of σ, where $l = \max_{i \in [\![1,p]\!]} cc(G[X_i])$, which proves the lemma.

For clarity in the description of σ, we denote by $\sigma_i + \sigma_j$ the concatenation of the orders σ_i and σ_j, which allows us to use the sum notation $\sum_{i=1 \text{ to } p} \sigma_i = \sigma_1 + \sigma_2 + \ldots + \sigma_p$. Beware that in the case of the concatenation operation, the sum is not commutative: it must be done in the specified order, from 1 to p. Then the order σ we build is simply defined as (see Figure 3):

$$\sigma = \sum_{1 \leq i \leq p,\ T_i \text{ is black}} \sigma_i + \sum_{1 \leq j \leq p,\ T_j \text{ is white}} \sigma_j.$$

We now prove that, in σ, the closed neighborhood $N[x]$ of any vertex $x \in V(G)$ is split in at most $l+2$ intervals. We separate the case where x is a leaf of some white T_i from the case where x is a leaf of some black T_i.

In the former case, the neighbors of x that are not in T_i are exactly those vertices belonging to some black X_j such that $j > i$. Then, we have $N[x] = (N[x] \cap X_i) \cup \bigcup_{j > i \text{ and } X_j \text{ is black}} X_j$. As σ_i realizes the closed contiguity of $G[X_i]$, which is by definition at most l, $N[x] \cap X_i$ is split in at most

l intervals in σ_i. And since $\bigcup_{j>i \text{ and } X_j \text{ is black}} X_j$ is an interval of σ, it follows that $N[x]$ is split in at most $l+1$ intervals in σ.
Now consider the case where x is a leaf of some black T_i. The neighbors of x that are not in T_i are exactly those vertices belonging to some black X_j, $j \neq i$, or belonging to some white $X_{j'}$ such that $j' < i$. Then, we have $N[x] = (N[x] \cap X_i) \cup (\bigcup_{j<i,\ X_j \text{ black}} X_j) \cup (\bigcup_{j>i,\ X_j \text{ black}} X_j \cup \bigcup_{j'<i,\ X_{j'} \text{ white}} X_{j'})$. Again, from the definition of σ_i, $N[x] \cap X_i$ is split in at most l intervals in σ_i. Moreover, $\bigcup_{j<i,\ X_j \text{ black}} X_j$ and $\bigcup_{j>i,\ X_j \text{ black}} X_j \cup \bigcup_{j'<i,\ X_{j'} \text{ white}} X_{j'}$ are two intervals of σ. Thus, $N[x]$ is split in at most $l+2$ intervals in σ. As this holds for any vertex x in G, it follows that $cc(G) \leq 2+l$, which ends the proof of the lemma. □

To give a detailed proof of Theorem 3.1, we actually start by proving the following Lemma.

Lemma 5.4. *Given a rooted tree T such that $rank(T) = k \geq 1$, there exists a root-path decomposition $\{T_1, \ldots, T_p\}$ of T such that for each $i \in [\![1, p]\!]$, $rank(T_i) \leq k-1$.*

Proof: Since T has rank k, from Theorem 2.1, T has path-height k. Consider a path partition $\mathcal{P}$ of T of height k where the path containing the root r of T is denoted by P_r. If the lowest node of P_r is a leaf, removing it from P_r and letting it form its own path still gives a path partition of height k, since $k \geq 1$. We thereby assume that P_r contains only internal nodes of T. We denote by $u_1, \ldots, u_p$ the nodes of T whose parent is in P_r. Clearly, for each $i \in [\![1, p]\!]$, the paths of $\mathcal{P}$ that contain some node of T_{u_i} form a path partition $\mathcal{P}_i$ of T_{u_i}. And since $\mathcal{P}$ has height k, then for all $i \in [\![1, p]\!]$, $\mathcal{P}_i$ has height at most $k-1$. It follows from Theorem 2.1 that $rank(T_{u_i}) \leq k-1$. Thus, $\{T_{u_1}, \ldots, T_{u_p}\}$ is a suitable root-path decomposition of T, which achieves the proof of the lemma. □

Theorem 5.3. *For any cograph G and its cotree T, $cc(G) \leq 2\,rank(T) + 1$.*

Proof: We prove this property by induction on the rank k of T.

For $k = 1$, we have to show that $cc(G) \leq 3$. Since T is a cotree, all its internal nodes have at least two children. It follows, since T has rank 1, that all its internal nodes are ancestors of each other, otherwise T would have rank at least 2. Thus, T is a rooted caterpillar, i.e. a rooted tree where all internal nodes are ancestors of the lowest nodes of the tree. Let us denote $V(G) = \{x_i\}_{1 \leq i \leq n}$. Since T is a caterpillar, the family $\{T_i\}_{1 \leq i \leq n}$, where T_i is the trivial tree formed with the single leaf x_i, is a root-path decomposition of T. With the notations of Lemma 3.1, the set of leaves of T_i is $X_i = \{x_i\}$. And since for any $i \in [\![1, n]\!]$, $cc(G[X_i]) \leq 1$, Lemma 3.1 concludes that $cc(G) \leq 2 + 1 = 3$.

Now, let $k \geq 1$ such that the property holds for all $k' \in [\![1, k]\!]$, that is any cograph G whose cotree T has rank k' satisfies $cc(G) \leq 2\,rank(T) + 1$. We prove this also holds for $k+1$. Let T be the cotree of a cograph G such that $rank(T) = k+1$. From Lemma 5.4, there exists a root-path decomposition $\{T_1, \ldots, T_p\}$ of T such that for all $i \in [\![1, p]\!]$, $rank(T_i) \leq k$. Thus, by applying the induction hypothesis on each T_i, we know that $\forall i \in [\![1, p]\!], cc(G[X_i]) \leq 2\,rank(T_i) + 1 \leq 2k+1$, where X_i is the set of leaves of T_i. By applying Lemma 3.1, we deduce that $cc(G) \leq 2 + \max_{i \in [\![1,p]\!]} cc(G[X_i]) \leq 2k+3 = 2(k+1)+1$. This ends the induction and the proof of the Lemma. □

Theorem 5.4. *The closed contiguity of a cograph is at most logarithmic in its number of vertices, or more formally, if $G = (V, E)$ is a cograph, then $cc(G) \leq 2\log_2 |V| + 1$.*

Proof: Let us denote by k the rank of the cotree T of G. From the definition of the rank, we deduce that T has at least 2^k leaves, which gives $|V| \geq 2^k$. It follows that $k \leq \log_2 |V|$. And since from Lemma 3.1, $cc(G) \leq 2k+1$, we conclude that $cc(G) \leq 2\log_2 |V| + 1$. □

Appendix C) A lower bound for the contiguity of cographs

Lemma 5.5. *For any cograph G, whose bicolored cotree has a white root and black rank at least $2k$, we have $cc(G) \geq k+1$.*

Sketch of proof. We prove it by induction. The result is obvious for $k = 1$.

Now, let us fix $k \geq 1$, and consider a cograph G whose bicolored cotree T has black rank at least $2(k+1)$. Then it contains the complete binary tree T_B of height $2(k+1)$, and each vertex of T_B can be mapped to one vertex of T: as T_B is a minor of T, among all vertices of T which were contracted into one vertex y of T_B, we choose one (also colored black), and also name it y.

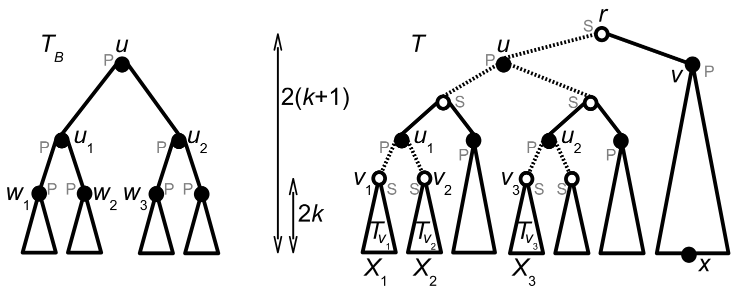

Fig. 6. The cotree T of black rank $2(k+1)$ used in the proof of Lemma 4.2, and the associated black complete binary tree T_B of height $2(k+1)$.

Thus, we label some vertices of T_B as illustrated in Figure 6: u is the root and its children are called u_1 and u_2. The two children of u_1 are called w_1 and w_2 and one child of u_2 is called w_3. We give the same labels to the mapped vertices in T, and add the following ones: r is the white root of T, v is a black child of r which is not an ancestor of u, and v_1, v_2, v_3 are the respective white parents of w_1, w_2, w_3. Finally, we denote by X_1, X_2, X_3 the set of leaves of respectively $T_{v_1}, T_{v_2}, T_{v_3}$ and by x be a leaf of T_v.

Vertex x is adjacent in G to all the vertices of $X_1 \cup X_2 \cup X_3$, and for any distinct i, j in $[\![1, 3]\!]$, all the vertices of X_i are non-adjacent to all the vertices of X_j. Moreover, $T_{v_1}, T_{v_2}, T_{v_3}$ are trees with a white root, and black rank at least $2k$. So we apply the induction hypothesis, and $cc(G[X_i]) \geq k$ for any $i \in [\![1, 3]\!]$.

Consider a linear order σ on the vertices of G. We denote by a (resp. b) the vertex of $X_1 \cup X_2 \cup X_3$ which is closer to x on the left resp. on the right) in σ, if it exists (i.e. if there is some node of $X_1 \cup X_2 \cup X_3$ on the left (resp. right) of x in σ). Clearly, there exists $i \in [\![1, 3]\!]$ such that none of a, b is in X_i. Then, for all the vertices $y \in X_i$, there exists some vertex $z \in \{a, b\}$ such that z is between x and y in σ, and z is not adjacent to y. Since X_i has contiguity at least k, there exists some vertex $y_1 \in X_i$ such that its neighborhood in $G[X_i]$ is split in at least k intervals in σ. Furthermore, since y_1 is adjacent to x but all the vertices of X_i are separated from x in σ by some $z \in \{a, b\}$ which is not adjacent to y_1, then it follows that the neighborhood of y_1 is split in at least $k+1$ intervals in σ. Thus, $cc(G) \geq k+1$, which ends the proof. □