Tree-Layout Based Graph Classes: Proper Chordal Graphs

Christophe Paul, Evangelos Protopapas

Tree-layout based graph classes:
proper chordal graphs[*]

Christophe Paul[†]   Evangelos Protopapas [‡]

July 3, 2024

Abstract

Many standard graph classes are known to be characterized by means of layouts (a vertex ordering) excluding some patterns. Important such graph classes are among others: proper interval graphs, interval graphs, chordal graphs, permutation graphs, (co-)comparability graphs. For example, a graph $G = (V, E)$ is a proper interval graph if and only if G has a layout $\mathbf{L}_G$ such that for every triple of vertices such that $x \prec_{\mathbf{L}_G} y \prec_{\mathbf{L}_G} z$, if $xz \in E$, then $xy \in E$ and $yz \in E$. Such a triple x, y, z is called an *indifference triple* and layouts excluding indifference triples are known as *indifference layouts*.

In this paper, we investigate the concept of *tree-layouts*. A tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ of a graph $G = (V, E)$ is a tree T rooted at some node r and equipped with a one-to-one mapping ρ_G between V and the nodes of T such that for every edge $xy \in E$, either x is an ancestor of y, denoted $x \prec_{\mathbf{T}_G} y$, or y is an ancestor of x. Clearly, layouts are tree-layouts.

Excluding a pattern in a tree-layout is defined similarly as excluding a pattern in a layout, but now using the ancestor relation. Unexplored graph classes can be defined by means of tree-layouts excluding some patterns. As a proof of concept, we show that excluding non-indifference triples in tree-layouts yields a natural notion of *proper chordal graphs*. We characterize proper chordal graphs and position them in the hierarchy of known subclasses of chordal graphs. We also provide a canonical representation of proper chordal graphs that encodes all the indifference tree-layouts rooted at some vertex. Based on this result, we first design a polynomial time recognition algorithm for proper chordal graphs. We then show that the problem of testing isomorphism between two proper chordal graphs is in P, whereas this problem is known to be GI-complete on chordal graphs.

[†]LIRMM, Univ Montpellier, CNRS, Montpellier, France.
[‡]LIRMM, Univ Montpellier, CNRS, Montpellier, France.

1 Introduction

Context. A graph class $\mathcal{C}$ is hereditary if for every graph $G \in \mathcal{C}$ and every induced subgraph H of G, which we denote $H \subseteq_i G$, we have that $H \in \mathcal{C}$. A *minimal forbidden subgraph* for $\mathcal{C}$ is a graph $F \notin \mathcal{C}$ such that for every induced subgraph $H \subseteq_i F$, $H \in \mathcal{C}$. Clearly, a hereditary graph class $\mathcal{C}$ is characterized by its set of minimal forbidden subgraphs. Let $\mathcal{F}$ be a set of graphs that are pairwise not induced subgraphs of one another. We say that a graph G is an $\mathcal{F}$-*free graph*, if it does not contain any graph of $\mathcal{F}$ as an induced subgraph. Many graph classes are characterized by a finite set $\mathcal{F}$ of minimal forbidden subgraphs. If $\mathcal{F} = \{H\}$, then we simply say that G is H-free if $H \nsubseteq_i G$. A popular example of such a class is the set of cographs [33, 46]. A graph G is a *cograph* if either G is the single vertex graph, or it is the disjoint union of two cographs, or its complement is a cograph. It is well known that G is a cograph if and only if it is a P_4-free graph [33, 10]. Unfortunately, the set of minimal obstructions of a hereditary graph family may not be finite. This is the case for *chordal graphs* [27, 3] which are defined as the graphs that do not contain a chordless cycle of length at least 4 as an induced subgraph.

An interesting approach, initiated by Skrien [45] and Damaschke [12], to circumvent this issue, is to embed graphs in some additional structure such as vertex orderings, also called *layouts*. An *ordered graph* is then defined as a pair $(G, \prec_G)$ such that $\prec_G$ is a total ordering of the vertex set V of the graph $G = (V, E)$. We say that an ordered graph $(H, \prec_H)$ is a *pattern* of the ordered graph $(G, \prec_G)$, which we denote by $(H, \prec_H) \subseteq_p (G, \prec_G)$, if $H \subseteq_i G$ and for every pair of vertices x and y of H, $x \prec_G y$ if and only if $x \prec_H y$. A graph G excludes the pattern $(H, \prec_H)$, if there exists a layout $\prec_G$ of G such that $(H, \prec_H) \nsubseteq_p (G, \prec_G)$. More generally, a graph class $\mathcal{C}$ excludes a set $\mathcal{P}$ of patterns if for every graph $G \in \mathcal{C}$, there exists a layout $\prec_G$ such that for every pattern $(H, \prec_H) \in \mathcal{P}$, $(H, \prec_H) \nsubseteq_p (G, \prec_G)$. We let $\mathfrak{L}(\mathcal{P})$ denote the class of graphs excluding a pattern from $\mathcal{P}$. Hereafter, a small size pattern $(H, \prec_H)$ will be encoded by listing its set of (ordered) edges and non-edges. There are two patterns on two vertices and eight patterns on three vertices, see Figure 1.

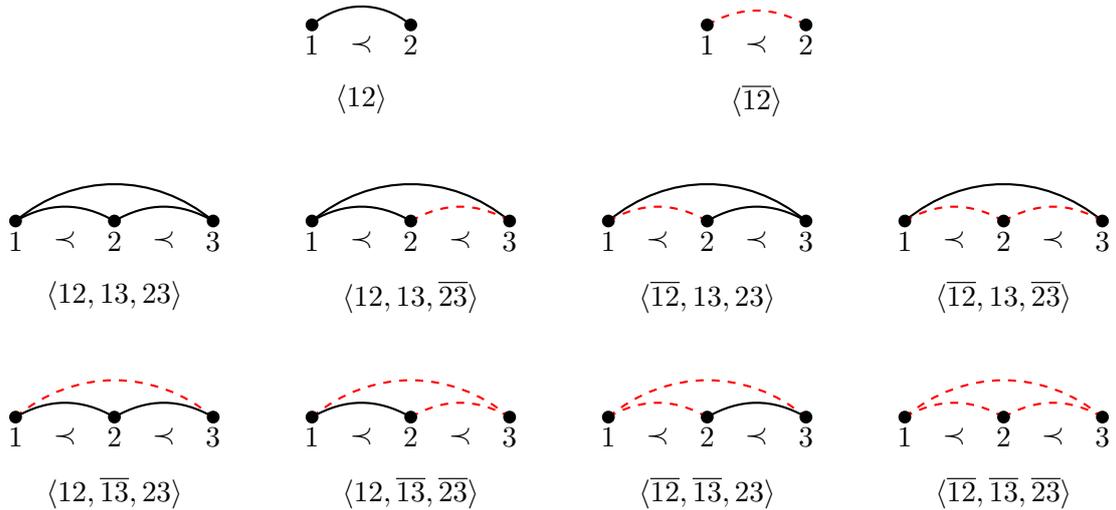

Figure 1: The patterns on at most 3 vertices. $\mathfrak{L}(\langle \overline{12}, 13, 23 \rangle)$ is the class of chordal graphs.

Interestingly, it is known that chordal graphs are characterized by excluding $\mathcal{P}_{\mathsf{chordal}} =$

$\{\langle\overline{12}, 13, 23\rangle\}$, see Figure 1 [12, 14]. This characterization relies on the fact that a graph is chordal if and only if it admits a *simplicial elimination ordering* [13, 44]. A vertex is *simplicial* if its neighbourhood induces a clique. A simplicial elimination ordering can be defined by a layout $\prec_G$ of $G = (V, E)$ such that every vertex x is simplicial in the subgraph $G[\{y \in V \mid y \prec_G x\}]$. Observe that a vertex is simplicial if and only if it is not the mid vertex of a P_3, the induced path on three vertices implies the excluded pattern characterization.

In fact, Ginn [21] prove that for every pattern $(H, \prec_H)$ such that H is neither the complete graph nor the edge-less graph, characterizing the graph family $\mathcal{L}((H, \prec_H))$ requires an infinite family of forbidden induced subgraphs. Observe however that excluding a unique pattern is important for that result. Indeed, cographs are characterized as P_4-free graphs (see discussion above) and need a set $\mathcal{P}_{\mathsf{cograph}}$ of several excluded patterns (see Figure 2) to be characterized [12]:

$$\mathcal{P}_{\mathsf{cograph}} = \left\{ \langle 12, \overline{13}, 23\rangle, \langle\overline{12}, 13, \overline{23}\rangle, \langle\overline{12}, 13, \overline{14}, 23, 24, \overline{34}\rangle, \langle 12, \overline{13}, 14, \overline{23}, \overline{24}, 34\rangle \right\}$$

$$\langle\overline{12}, 13, \overline{14}, 23, 24, \overline{34}\rangle \qquad\qquad \langle 12, \overline{13}, 14, \overline{23}, \overline{24}, 34\rangle$$

Figure 2: The two size 4 forbidden patterns of cographs.

In [15], Duffus et al. investigate the computational complexity of the recognition problem of $\mathcal{L}((H, \prec_H))$ for a fixed ordered graph $(H, \prec_H)$. They conjectured that if H is neither the complete graph nor the edge-less graph, then recognizing $\mathcal{L}((H, \prec_H))$ is NP-complete if H or its complement is 2-connected. Hell et al. [29] have recently shown that if $\mathcal{P}$ only contains patterns of size at most 3, then $\mathfrak{L}(\mathcal{P})$ can be recognized in polynomial time using a 2-SAT approach. Besides chordal graphs (see discussion above), these graph classes comprise very important graph classes, among others:

- Bipartite graphs exclude $\mathcal{P}_{\mathsf{bip}} = \{\langle 12, 13, 23\rangle, \langle 12, \overline{13}, 23\rangle\}$: If $B = (X, Y, E)$, then every layout such that for every $x \in X$, $y \in Y$, $x \prec_B y$ is $\mathcal{P}_{\mathsf{bip}}$-free.

- Forests exclude $\mathcal{P}_{\mathsf{forest}} = \{\langle 12, 13, 23\rangle, \langle\overline{12}, 13, 23\rangle\}$: A $\mathcal{P}_{\mathsf{forest}}$-free layout is obtained by ordering the vertices of T according to a depth first search ordering of T.

- *Co-comparability graphs* [20, 18] exclude $\mathcal{P}_{\mathsf{cocomp}} = \{\langle\overline{12}, 13, \overline{23}\rangle\}$: A co-comparability graph is a graph whose complement can be transitively oriented. A $\mathcal{P}_{\mathsf{cocomp}}$-free layout is obtained as a linear extension of a transitive orientation of $\overline{G}$;

- *Interval graphs* [28, 2, 23] exclude $\mathcal{P}_{\mathsf{int}} = \mathcal{P}_{\mathsf{cocomp}} \cup \mathcal{P}_{\mathsf{chordal}}$ [41]: A graph is an interval graph if it is the intersection graph of a set of intervals on the real line. The existence of a $\mathcal{P}_{\mathsf{int}}$-layout for interval graphs follows from the fact that a graph is an interval graph if and only if it is chordal and co-comparability.

- *Proper interval graphs* [42, 43] exclude $\mathcal{P}_{\mathsf{proper}} = \{\langle 12, 13, \overline{23}\rangle, \langle\overline{12}, 13, 23\rangle\}$ [12]. A graph is a proper interval graph if it is the intersection graph of a set of proper intervals on the real line (no interval is a subset of another one). We observe that proper interval

graphs where originally characterized by the existence of a so-called *indifference orderings* [42, 43] that are exactly the layouts excluding $\mathcal{P}_{\text{int}} \cup \{\langle 12, 13, \overline{23} \rangle\}$ [42, 43]. As we will see later, a layout excluding $\mathcal{P}_{\text{int}} \cup \{\langle 12, 13, \overline{23} \rangle\}$ is a $\mathcal{P}_{\text{proper}}$-free layout.

- *Trivially perfect graphs* [22] exclude $\mathcal{P}_{\text{trivPer}} = \mathcal{P}_{\text{chordal}} \cup \mathcal{P}_{\text{comp}}$. A graph G is a trivially perfect graph if and only if it is $\{P_4, C_4\}$-free, or equivalently G is the comparability graph of a rooted tree T (two vertices are adjacent if one is the ancestor of the other). A $\mathcal{P}_{\text{trivPer}}$-free layout is obtained from a depth first search ordering of T. Moreover every layout of the P_4 and the C_4 contains one of the patterns of $\mathcal{P}_{\text{trivPer}}$ (see [17]).

For more examples, the reader should refer to [12, 17]. Feuilloley and Habib [17] list all the graph classes that can be obtained by excluding a set of patterns each of size at most 3. Moreover, for most of them (but two), they argue about the existence of a linear time recognition algorithm.

From layouts to tree-layouts. A layout $\prec_G$ of a graph $G = (V, E)$ on n vertices can be viewed as an embedding of G into a path P on n vertices rooted at one of its extremities. Under this view point, it becomes natural to consider graph embeddings in graphs that are more general than rooted paths. Recently, Guzman-Pro et al. [24] have studied embeddings in a cyclic ordering. In this paper, we consider embedding the vertices of a graph in a rooted tree, yielding the notion of *tree-layout*.

Definition 1. *Let $G = (V, E)$ be a graph on n vertices. A* tree-layout *of G is a triple $\mathbf{T}_G = (T, r, \rho_G)$ where T is a tree on a set V_T of n nodes rooted at r and $\rho_G : V \to V_T$ is a bijection such that for every edge $xy \in E$, either x is an ancestor of y, denoted by $x \prec_{\mathbf{T}} y$, or y is an ancestor of x.*
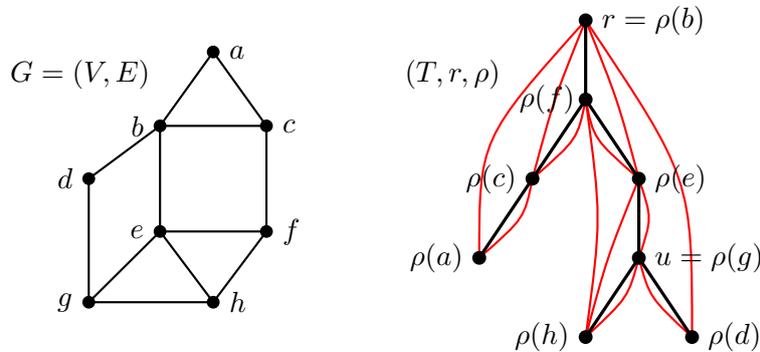
Figure 3: A tree-layout (T, r, ρ) of a graph $G = (V, E)$.

We observe that, from Definition 1, in a *tree-layout* $\mathbf{T}_G = (T, r, \rho_G)$ of a graph G, T is not a *Trémaux tree* since it is not necessarily a spanning tree of G (see [40] and [6] for similar concepts). However, it is easy to see that a tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ such that T is a path is a layout of G. So from now on, we shall define a *layout* as a triple $\mathbf{L}_H = (P, r, \rho_H)$, where P is a path that fulfils the conditions of Definition 1. An ordered graph then becomes a pair $(H, \mathbf{L}_H)$ where $\mathbf{L}_H = (P, r, \rho_H)$ is a layout of G. Excluding a pattern $(H, \mathbf{L}_H)$ in a tree-layout of a graph G is defined similarly as excluding a pattern in a layout, but now using the ancestor relation.

4

For a set $\mathcal{P}$ of patterns, we can also define the class $\mathfrak{T}(\mathcal{P})$ of graphs admitting a tree-layout that excludes every pattern $P \in \mathcal{P}$. If $\mathcal{P} = \{(H, \mathbf{L}_H)\}$, we simply write $\mathcal{T}((H, \mathbf{L}_H))$. Observe that, as a layout is a tree-layout, for a fixed set $\mathcal{P}$ of patterns, we always have $\mathfrak{L}(\mathcal{P}) \subseteq \mathfrak{T}(\mathcal{P})$. As an introductory example, let us consider the pattern $\langle\overline{12}\rangle$. The following observation directly follows from the definitions of a tree-layout and trivially perfect graphs. Indeed, recall that a trivially perfect graph G is the comparability graph of a rooted tree T, that is two vertices are adjacent if one is the ancestor of the other. It follows that if r is the root of T, then $(T, r, \mathbb{I})$, where $\mathbb{I}$ is the identity, is a tree-layout of G.

Observation 1. *The class $\mathcal{L}(\langle\overline{12}\rangle)$ is the set of complete graphs while the class $\mathcal{T}(\langle\overline{12}\rangle)$ is the set of trivially perfect graphs.*

So the class of trivially perfect graphs can be viewed as the tree-like version of the class of complete graphs. This view point motivates the systematic study of graph classes defined by excluding a fixed pattern (or a fixed set of patterns) in a tree-layout. The first questions are probably to understand the graph classes we obtain by excluding patterns of size at most 3 in tree-layouts. Do we retrieve some known graph classes or do we define novel graph classes? What is the computational complexity of the recognition problem of these graph classes? What are the relationship of these novel graph classes with the known ones? Do these novel graph classes allows to solve in polynomial time problems that are NP-hard on arbitrary graphs?

Our contributions. As a first study of tree-layout based graph classes, we consider the patterns characterizing interval graphs and proper interval graphs. We first show that if we consider the interval graphs patterns $\mathcal{P}_{\mathsf{int}}$, the same phenomena as for $\{\langle\overline{12}\rangle\}$ holds, leading to a novel (up to our knowledge) characterization of chordal graphs as being exactly $\mathfrak{T}(\mathcal{P}_{\mathsf{int}})$ (see Theorem 3).

As already discussed, proper interval graphs are obtained by restricting interval graphs to the intersection of a set of *proper* intervals (no interval is a subinterval of another). This restriction leads to a distinct graph class as the $K_{1,3}$ is an interval graph but not a proper one. Following this line, in his seminal paper [19] characterizing chordal graphs as the intersection graphs of a subset of subtrees of a tree, Gavril considered the class of intersection graphs of a set of *proper* subtrees of a tree (no subtree is contained in an another). Using an easy reduction, Gavril proved that this again yields a characterization of chordal graphs. So this left open the question of proposing a natural definition for proper chordal graphs, a class of graphs that should be sandwiched between proper interval graphs and chordal graphs but incomparable to interval graphs. Our main contribution is to propose such a natural definition of *proper chordal graphs* by means of forbidden patterns on tree-layouts: a graph is proper chordal if it belongs to $\mathfrak{T}(\mathcal{P}_{\mathsf{proper}})$, the class of graphs admitting a $\mathcal{P}_{\mathsf{proper}}$-free tree-layout. Recall that $\mathcal{P}_{\mathsf{proper}}$ are the patterns characterizing proper interval graphs on layouts. Table 1 resumes the discussion above about graph classes respectively obtained from layouts and tree-layout excluding a fixed set of patterns and positions proper chordal graphs with respect to trivially perfect graphs, chordal graphs and proper interval graphs. In a recent paper, Chaplick [8] investigated this question and considered the class of intersection graphs of non-crossing paths in a tree.

We then provide a thorough study of the class of proper chordal graphs, both on their combinatorial as well as their algorithmic aspects. In reference to the *indifference layout* characterizing proper interval graphs, we call a $\mathcal{P}_{\mathsf{proper}}$-free tree-layout an *indifference tree-layout*.

Forbidden patterns	Layouts	Tree-layouts
$\langle\overline{12}\rangle$	Cliques	Trivially perfect graphs
$\langle 12, 13, \overline{23}\rangle, \langle\overline{12}, 13, 23\rangle$	Proper interval graphs	**Proper chordal graphs**
$\langle\overline{12}, 13, \overline{23}\rangle, \langle\overline{12}, 13, 23\rangle$	Interval graphs	Chordal graphs

Table 1: Graph classes obtained by excluding $\langle\overline{12}\rangle$, $\mathcal{P}_{\mathsf{proper}}$ and $\mathcal{P}_{\mathsf{int}}$.

Our first result (see Theorem 6) is a characterization of indifference tree-layouts (and henceforth of proper chordal graphs). As discussed in Section 3, (proper) interval graphs have, in general, multiple (proper) interval representations, which are all captured by a canonical tree structure called PQ-tree [7]. We show that the set of indifference tree-layouts can also be represented in a canonical and compact way, see Theorem 7. These structural results have very interesting algorithmic implications. First, we can design a polynomial time recognition algorithm for proper chordal graphs, see Theorem 8. Second, we show that the isomorphism problem restricted to proper chordal graphs is polynomial time solvable, see Theorem 9. Interestingly, this problem is GI-complete on (strongly) chordal graphs [39]. So considering proper chordal graphs allows us to push the tractability further towards its limit.

2 Preliminaries

2.1 Notations and definitions

Graphs. In this paper, every graph is finite, loopless, and without multiple edges. A graph is a pair $G = (V, E)$ where V is its vertex set and $E \subseteq V^2$ is the set of edges. For two vertices $x, y \in V$, we let xy denoted the edge $e = \{x, y\}$. We say that the vertices x and y are incident with the edge xy. The neighbourhood of a vertex x is the set of vertices $N(x) = \{y \in V \mid xy \in E\}$. Let S be a subset of vertices of V. The closed neighbourhood of a vertex x is $N[x] = N(x) \cup \{x\}$. The graph resulting from the removal of a vertex subset S of $V(G)$ is denoted by $G - S$. If $S = \{x\}$ is a singleton we write $G - x$ instead of $G - \{x\}$. The subgraph of G induced by S is $G[S] = G - (V \setminus S)$. We say that S is a *separator* of G if $G - S$ contains more connected components than G. We say that S separates $X \subseteq V(G)$ from $Y \subseteq V(G)$ if X and Y are subsets of distinct connected components of $G - S$. If x is a vertex such that $x \notin S$ and $S \subseteq N(x)$, then we say that x is *S-universal.*

Rooted trees. A rooted tree is a pair (T, r) where r is a distinguished node[1] of the tree T. We say that a node u is an *ancestor* of the node v (and that u is a *descendant* of v) if u belongs to the unique path of T from r to v. If u is an ancestor of v, then we write $u \prec_{(T,r)} v$. For a node u of T, the set $A_{(T,r)}(u)$ contains every *ancestor* of u, that is every node v of T such that $v \prec_{(T,r)} u$. Likewise, the set $D_{(T,r)}(u)$ contains every *descendant* of u, that is every node v of T such that $u \prec_{(T,r)} v$. We may also write $A_{(T,r)}[u] = A_{(T,r)}(u) \cup \{u\}$ and $D_{(T,r)}[u] = D_{(T,r)}(u) \cup \{u\}$. The least common ancestor of two nodes u and v is denoted $\mathsf{lca}_{(T,r)}(u, v)$. For a node u, we define T_u as the subtree of (T, r) rooted at u and containing the descendants of u. We let $\mathcal{L}(T, r)$ denote the set of leaves of (T, r) and for a node u, $\mathcal{L}_{(T,r)}(u)$ is the set of leaves of (T, r) that are descendants of the node u.

---

[1] To avoid confusion, we reserve the term vertex for graphs and nodes for trees.

Ordered trees. An ordered tree is a rooted tree (T, r) such that the children of every internal node are totally ordered. For an internal node v, we let denote $\sigma^v_{(T,r)}$ the permutation of its children. An ordered tree is non-trivial if it contains at least one internal node. An ordered tree (T, r) defines a permutation $\sigma_{(T,r)}$ of its leaf set $\mathcal{L}(T, r)$ as follows. For every pair of leaves $x, y \in \mathcal{L}(T, r)$, let u_x and u_y be the children of $v = \mathbf{lca}_{(T,r)}(x, y)$ respectively being an ancestor of x and of y. Then $x \prec_{\sigma_{(T,r)}} y$ if and only if $u_x \prec_{\sigma^v_{T,r}} u_y$.

Tree-layouts of graphs. Let $\mathbf{T} = (T, r, \rho)$ be a tree-layout of a graph $G = (V, E)$ (see Definition 1). Let us recall that from Definition 1, G is a subgraph of the transitive closure of (T, r). Let x and y be two vertices of G. We note $x \prec_{\mathbf{T}} y$ if $\rho(x)$ is an ancestor of $\rho(y)$ in (T, r) and use $A_{\mathbf{T}}(x)$, $D_{\mathbf{T}}(x)$ to respectively denote the ancestors and descendants of x. The notations $\mathcal{L}(\mathbf{T})$, $\mathbf{T}_x$, $\mathcal{L}_{\mathbf{T}}(x)$ are derived from the notations defined above.

2.2 Chordal and interval graphs

Let us discuss in further details the classes of chordal, interval and proper interval graphs. A graph is *chordal* if it does not contain any cycle C_k with $k \geq 4$ as induced subgraph (or equivalently every cycle of length at least 4 contains a chord) [27, 3]. Gavril [19] characterized chordal graphs as the intersection graphs of a family of subtrees of a tree. Given a chordal graph $G = (V, E)$, a *tree-intersection model* of $G = (V, E)$ is defined as a triple $\mathbf{M}^{\mathsf{T}}_G = (T, \mathcal{T}, \tau_G)$ where T is a tree, $\mathcal{T}$ is a family of subtrees of T and $\tau_G : V \to \mathcal{T}$ is a bijection such that $xy \in E$ if and only if $\tau_G(x)$ intersects $\tau_G(y)$. Hereafter, we denote by $T^x \in \mathcal{T}$ the subtree of T such that $T^x = \tau_G(x)$. Since chordal graphs are characterized by the existence of a simplicial elimination ordering [44], we obtain the following.

Theorem 1. *[12] The class of chordal graphs is $\mathfrak{L}(\mathcal{P}_{\mathsf{chordal}})$ where*
$$\mathcal{P}_{\mathsf{chordal}} = \left\{ \langle \overline{12}, 13, 23 \rangle \right\}.$$

A graph $G = (V, E)$ is an *interval graph* if it is the intersection graph of a family of intervals on a line [28, 2, 23]. So the intersection model of an interval graph G is $\mathbf{M}^{\mathsf{I}}_G = (P, \mathcal{P}, \tau_G)$ where P is a path, $\mathcal{P}$ is a family of subpaths of P. The interval, or subpath, $\tau_G(x) \in \mathcal{P}$ will be denoted P^x. Observe that if G contains n vertices then there exists $\mathbf{M}^{\mathsf{I}}_G = (P, \mathcal{P}, \tau_G)$ s.t. P has at most $2n$ nodes. It follows that for every $x \in V$, P^x can be represented by an interval $[l(x), r(x)]$. As compared to chordal graphs, we restrict the intersection model to paths, interval graphs form a subset of chordal graphs. Moreover, it is easy to observe that the intersection model of an interval naturally defines a transitive orientation of the non-edges of G. These observations lead to the following.

Theorem 2. *[12] The class of interval graphs is $\mathfrak{L}(\mathcal{P}_{\mathsf{int}})$ where*
$$\mathcal{P}_{\mathsf{int}} = \left\{ \langle \overline{12}, 13, 23 \rangle, \langle \overline{12}, 13, \overline{23} \rangle \right\}.$$

To sketch the proof of Theorem 2, we describe how a $\mathcal{P}_{\mathsf{int}}$-free layout of G can be obtained from an interval-intersection model of G and vice-versa. Suppose that $\mathbf{M}^{\mathsf{I}}_G$ is an interval-intersection model of the interval graph $G = (V, E)$. Then a $\mathcal{P}_{\mathsf{int}}$-free layout $\mathbf{L}_G = (P, r, \rho_G)$ of G is obtained as follows: for every $x, y \in V$, $x \prec_{\mathbf{L}_G} y$ if and only if $r(x) < r(y)$ or $l(x) < l(y) < r(y) < r(x)$ (see Figure 4).

Suppose that $\mathbf{L}_G$ is a $\mathcal{P}_{\mathsf{int}}$-free layout of G. For every vertex $x \in V$, we define $\sigma(x) = |A_{\mathbf{T}}(x)| + 1$. Then an interval-intersection model $\mathbf{M}^{\mathsf{I}}_G$ of G is obtained as follows: with every
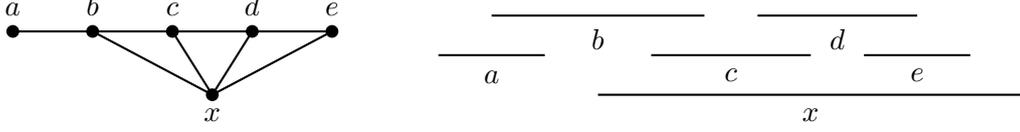
Figure 4: An interval graph $G = (V, E)$ and an interval representation of G. The layout $\mathbf{L}_G = (P, r, \rho)$ associated to that interval representation is $a \prec_{\mathbf{L}_G} b \prec_{\mathbf{L}_G} x \prec_{\mathbf{L}_G} c \prec_{\mathbf{L}_G} d \prec_{\mathbf{L}_G} e$.

vertex $x \in V$, we associate the interval $[\sigma(x), \sigma(y)]$ where y is the vertex in $N[x]$ such that $x \prec_{\mathbf{L}_G} y$ and for every vertex z such that $y \prec_{\mathbf{L}_G} z$, $z \notin N(x)$.

From the discussion above, it is legitimate to characterize chordal graphs as the "tree-like" interval graphs. Indeed, this is how the tree intersection model was introduced in the seminal paper of Gavril [19]. It is then natural to wonder whether the phenomena observed with the classes $\mathcal{L}(\langle \overline{12} \rangle)$ and $\mathcal{T}(\langle \overline{12} \rangle)$ (see the discussion in Section 1) is confirmed with the set of patterns $\mathcal{P}_{\mathsf{int}}$. Surprisingly this question yields (up to our knowledge), the following novel characterization of chordal graphs.

Theorem 3. *The class of chordal graphs is $\mathfrak{T}(\mathcal{P}_{\mathsf{int}})$.*

Proof. The proof is based on a construction similar to the one we sketched for Theorem 2.
$\Rightarrow$ Let $\mathbf{M}_G^{\mathsf{T}} = (T', \mathcal{T}', \tau_G)$ be a tree-intersection model of G. We construct a tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ of G as follows. First we root T' at an arbitrary node r. This defines for every vertex x, a node $\rho'(x)$ which is the root of the subtree $\tau(x)$ of T'. Observe that we may assume that r is the root $\rho'(x)$ for some vertex $x \in V$. We set $R(T') = \{u \in V_{T'} \mid \exists x \in V, \rho'(x) = u\}$. The tree T is obtained from T' by first contracting every tree-edge uv such that v the child of u does not belong to $R(T')$. These contractions preserves the mapping ρ'. Then every node u of the resulting tree is replaced by a path P_u of length $\ell = |\{x \in V \mid u = \rho(x)\}|$. Finally, ρ' is modified to turn it into a bijection ρ_G from V to V_T as follows: the vertices mapped to a given node u of T' are mapped to distinct nodes of P_u (see Figure 5). Let us prove that the construction is correct.
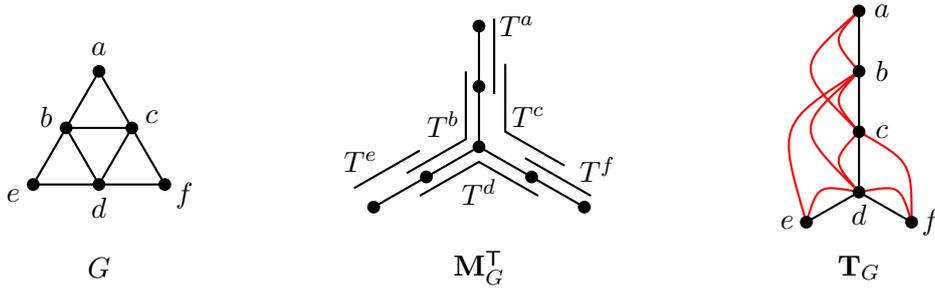
Figure 5: The 3-sun graph G on the left, a tree intersection model $\mathbf{M}_G^{\mathsf{T}}$ of G in the center and a $\mathcal{P}_{\mathsf{int}}$-free tree-layout of G on the right.

We first argue that $\mathbf{T}_G = (T, r, \rho_G)$ is a tree-layout of G, that is for every edge $xy \in E$, either $x \prec_{\mathbf{T}_G} y$ or $y \prec_{\mathbf{T}_G} x$. To see this, it suffices to observe that as the subtree T^x intersects the subtree T^y, then either $\rho_G(x)$ is an ancestor of $\rho_G(y)$ in T' or vice-versa.

Consider three vertices x, y and z such that $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$ and suppose that $xz \in E$. The fact that $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$ implies that either $\rho'(y)$ is a descendant in T' (when rooted at r) of $\rho'(x)$ or $\rho'(x) = \rho'(y)$ and that either $\rho'(y)$ is an ancestor of $\rho'(z)$ or $\rho'(y) = \rho'(z)$. Since $xz \in E$, then the subtrees T^x and T^z intersect. This implies T^x contains the path of T' from $\rho'(x)$ to $\rho'(z)$, and thereby contains the node $\rho'(y)$. It follows that $xy \in E$, implying that the pattern on x, y and z is neither isomorphic to $\langle \overline{12}, 13, 23 \rangle$ nor to $\langle \overline{12}, 13, \overline{23} \rangle$.

$\Leftarrow$ Suppose that $\mathbf{T}_G = (T, r, \rho_G)$ is a $\mathcal{P}_{\mathsf{int}}$-free tree-layout of G. For every vertex x, we define the subset of vertices $\delta(x) = \{ y \in D_{\mathbf{T}}(x) \mid xy \in E \land \forall z, y \prec_{\mathbf{T}_G} z, xz \notin E \}$. Then we build a tree-intersection model $\mathbf{M}_G^{\mathsf{T}} = (T, \mathcal{T}, \tau_G)$ of G as follows: for every $x \in V$, $\tau_G(x)$ is the smallest subtree of T containing the vertices of $\delta(x) \cup \{x\}$. It is straightforward to see that if $xy \in E$ then $\tau_G(x)$ and $\tau_G(y)$ intersect. Suppose the $\tau_G(x)$ and $\tau_G(y)$ intersect. Without loss of generality we can assume that $\rho_G(x) \prec_{\mathbf{T}_G} \rho_G(y)$ and thereby $\rho_G(y)$ belongs to $\tau_G(x)$. For the sake of contradiction, suppose that $xy \notin E$. By definion of $\tau_G(x)$, there exists a vertex z such that $\rho_G(y) \prec_{\mathbf{T}_G} \rho_G(z)$ and $xz \in E$. In turn, by the condition of Theorem 3 we have that $xy \in E$: contradiction. $\qquad \square$

We conclude the discussion on chordal graphs by noticing the observation that chordal graphs are also characterized as $\mathfrak{T}(\mathcal{P}_{\mathsf{chordal}})$. This follows from Lemma 1 below and the observation that $\mathcal{P}_{\mathsf{chordal}} \subset \mathcal{P}_{\mathsf{int}}$. So we obtain that $\mathfrak{T}(\mathcal{P}_{\mathsf{chordal}}) = \mathfrak{T}(\mathcal{P}_{\mathsf{int}})$. Hence for a fixed set of forbidden patterns, moving from layouts to tree-layouts does not always produce a larger class of graphs.

Lemma 1. *Let G be a chordal graph. Every $\mathcal{P}_{\mathsf{chordal}}$-free tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ of G can be transformed into a $\mathcal{P}_{\mathsf{chordal}}$-free layout $\mathbf{L}_G = (P, r, \rho_G')$ of G.*

Proof. The mapping ρ_G' is obtained by a depth-first search ordering of the nodes of T. That is $x \prec_{\mathbf{L}_G} y$ if and only if $\rho_G(x)$ is visited before $\rho_G(y)$ in such an ordering. Now consider three vertices x, y and z such that $x \prec_{\mathbf{L}_G} y \prec_{\mathbf{L}_G} z$ and such that $xz \in E$. To prove that $\mathbf{L}_G$ is $\mathcal{P}_{\mathsf{chordal}}$-free, we show that if $yz \in E$, then $xy \in E$. As $xz \in E$, we have that $x \prec_{\mathbf{T}_G} z$. Similarly if $yz \in E$, then we have $y \prec_{\mathbf{T}_G} z$. If we also have $x \prec_{\mathbf{T}_G} y$, then since $\mathbf{T}_G$ is $\mathcal{P}_{\mathsf{chordal}}$-free, $yx \in E$. So suppose that $x \nprec_{\mathbf{T}_G} y$. But since $x \prec_{\mathbf{L}_G} y$, $\rho_G(y)$ is visited after $\rho_G(x)$ and all its descendents, including $\rho_G(y)$: a contradiction. $\qquad \square$

2.3 Proper interval graphs and proper chordal graphs

Proper interval graphs. A graph $G = (V, E)$ is a *proper interval graph* if it is an interval graph admitting an interval model $\mathbf{M}_G^{\mathsf{I}}$ such that for every pair of intervals none is a subinterval of another [42, 43]. Proper interval graphs are also characterized as $K_{1,3}$-free interval graphs or as *unit interval graphs*, that are interval graphs admitting an interval model $\mathbf{M}_G^{\mathsf{I}}$ in which every interval has the same length. Let us consider the following set of patterns:

$$\mathcal{P}_{\mathsf{indifference}} = \big\{ \langle \overline{12}, 13, 23 \rangle, \langle \overline{12}, 13, \overline{23} \rangle, \langle 12, 13, \overline{23} \rangle \big\}.$$

A $\mathcal{P}_{\mathsf{indifference}}$-free layout is called an *indifference layout*. Indifference layouts have several characterizations:

Theorem 4. *[36, 42] Let $\mathbf{L}_G$ be a layout of a graph G. The following properties are equivalent.*

1. $\mathbf{L}_G$ is an indifference layout;

2. *for every vertex v, $N[v]$ is consecutive in $\mathbf{L}_G$;*

3. *every maximal clique is consecutive in $\mathbf{L}_G$;*

4. *for every pair of vertices x and y with $x \prec_{\mathbf{L}_G} y$, $N(y) \cap A_{\mathbf{L}_G}(x) \subseteq N(x) \cap A_{\mathbf{L}_G}(x)$ and $N(x) \cap D_{\mathbf{L}_G}(y) \subseteq N(y) \cap D_{\mathbf{L}_G}(y)$.*

Moreover G has an indifference layout if and only if it is a proper interval graph.

The following lemma shows that $\mathcal{P}_{\mathsf{indifference}}$ is not a minimal set of patterns to characterize proper interval graphs, yielding a proof of Theorem 5 below.

Lemma 2. *Let $\mathbf{L}_G$ be a $\left\{ \langle \overline{12}, 13, 23 \rangle, \langle 12, 13, \overline{23} \rangle \right\}$-free layout of a connected graph G. Then $\mathbf{L}_G$ an indifference layout.*

Proof. For the sake of contradiction, suppose that $\mathbf{L}_G$ contains the pattern $\langle \overline{12}, 13, \overline{23} \rangle$. So there exists a triple x, y, z of vertices such that $x \prec_{\mathbf{L}_G} y \prec_{\mathbf{L}_G} z$, $xz \in E$, $xy \notin E$ and $yz \notin E$. Let us consider such a triple that minimizes the set $\{ v \in V \mid x \prec_{\mathbf{L}_G} v \prec_{\mathbf{L}_G} z \}$ and call it a *bad* triple.

Since G is connected, it contains a shortest x, y-path P. We first observe that every internal vertex v of P satisfies $x \prec_{\mathbf{L}_G} v \prec_{\mathbf{L}_G} y$. Let v be the leftmost vertex of P in $\mathbf{L}_G$. Assume that $v \prec_{\mathbf{L}_G} x$. Then v has two neighbors a and b in P and at least one of them is distinct from x and y. Since $\mathbf{L}_G$ excludes $\langle 12, 13, \overline{23} \rangle$, $v \prec_{\mathbf{L}_G} a$ and $v \prec_{\mathbf{L}_G} b$, a and b are adjacent vertices, contradicting P to be a shortest x, y-path. Thereby the leftmost vertex v of P in $\mathbf{L}_G$ is x. For a symmetric argument, since $\mathbf{L}_G$ excludes $\langle \overline{12}, 13, 23 \rangle$, the rightmost vertex of P is y.

Since $xy \notin E$, P contains at least one vertex. Consider v the unique neighbour of x on P, then $x \prec_{\mathbf{L}_G} v \prec_{\mathbf{L}_G} y$. Since $\mathbf{L}_G$ excludes $\langle 12, 13, \overline{23} \rangle$, we have $vz \in E$, which in turns implies that $vy \notin E$. But then since $yz \notin E$, the triple v, y, z forms a $\langle \overline{12}, 13, \overline{23} \rangle$ pattern. Since $x \prec_{\mathbf{L}_G} v \prec_{\mathbf{L}_G} y$, this contradicts the choice of x, y, z as a bad triple. $\square$

Theorem 5. *[12] The class of proper interval graphs is $\mathfrak{L}(\mathcal{P}_{\mathsf{proper}})$ where*

$$\mathcal{P}_{\mathsf{proper}} = \left\{ \langle \overline{12}, 13, 23 \rangle, \langle 12, 13, \overline{23} \rangle \right\}.$$

Proper chordal graphs. From the discussion of Subsection 2.2, we would like to understand what are the *"tree-like"* proper interval graphs. In [19], Gavril already considered this question, but didn't provide a conclusive answer. He observed that any family $\mathcal{T}$ of subtrees of a tree T can be transformed into a proper family $\mathcal{T}'$ (no subtree in $\mathcal{T}'$ is a subtree of another one) by adding a private pendant leaf to every subtree of $\mathcal{T}$. Clearly the intersection graph of $\mathcal{T}$ is the same as the one of $\mathcal{T}'$. It follows that considering a proper family of subtrees also characterizes chordal graphs. To cope with *"tree-like"* proper interval graphs, we rather propose the following definition.

Definition 2. *A graph $G = (V, E)$ is a proper chordal graph if $G \in \mathfrak{T}(\mathcal{P}_{\mathsf{proper}})$.*

As a consequence of the fact that $\mathcal{P}_{\mathsf{proper}} \subset \mathcal{P}_{\mathsf{indifference}}$ and of Lemma 2, one can observe that a $\mathfrak{T}(\mathcal{P}_{\mathsf{proper}}) = \mathfrak{T}(\mathcal{P}_{\mathsf{indifference}})$. Thereby hereafter, a $\mathcal{P}_{\mathsf{proper}}$-free tree-layout $\mathbf{T}_G$ of a graph G will be called an *indifference tree-layout*. We first prove that Theorem 4 generalizes to indifference tree-layouts.

Theorem 6. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be a tree-layout of a graph G. The following properties are equivalent.*

1. *$\mathbf{T}_G$ is an indifference tree-layout;*

2. *for every vertex x, the vertices of $N[x]$ induces a connected subtree of T;*

3. *for every maximal clique K, the vertices of K appear consecutively on a path from r in T;*

4. *for every pair of vertices x and y such that $x \prec_{\mathbf{T}_G} y$, $N(y) \cap A_{\mathbf{T}_G}(x) \subseteq N(x) \cap A_{\mathbf{T}_G}(x)$ and $N(x) \cap D_{\mathbf{T}_G}(y) \subseteq N(y) \cap D_{\mathbf{T}_G}(y)$.*

Proof. $(1 \Leftrightarrow 2)$ Suppose that $N[x]$ is not connected. Then, there exists $z \in N[x]$ such that the path from x to z in T contains a vertex $y \notin N[x]$. Observe that as $xz \in E$, we have either $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$ or $z \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} x$, with $xz \in E$ but $xy \notin E$. But then, $\mathbf{T}$ is not $\mathcal{P}_{\mathsf{proper}}$-free and hence not an indifference tree-layout.

If $\mathbf{T}_G$ is not an indifference tree-layout, then it contains a pattern of $\mathcal{P}_{\mathsf{proper}}$, namely three vertices x, y and z such that $x \prec_{\mathbf{T}} y \prec_{\mathbf{T}} z$, $xz \in E$ but either $xy \notin E$ or $yz \notin E$. In the former case, $N[x]$ is not connected in T, in the latter case, $N[z]$ is not connected in T.

$(1 \Leftrightarrow 3)$ Suppose that $\mathbf{T}_G$ is an indifference tree-layout. Let K be a maximal clique of G. First observe that if x and y are vertices of K, then either $x \prec_{\mathbf{T}} y$ or $y \prec_{\mathbf{T}} x$. So let x be the vertex of K that is the closest to the root r and y the vertex of K that is the furthest from r. Let z be a vertex of G such that $x \prec_{\mathbf{T}_G} z \prec_{\mathbf{T}_G} y$. Since $\mathbf{T}_G$ is a $\{\langle \overline{12}, 13, \overline{23} \rangle, \langle \overline{12}, 13, 23 \rangle, \langle 12, 13, \overline{23} \rangle\}$-free tree-layout (2), $xz \in E$ and $yz \in E$. Now suppose that there exists $z' \neq z$ such that $x \prec_{\mathbf{T}} z' \prec_{\mathbf{T}} y$, by the same argument we have $xz' \in E$ and $yz' \in E$. Since $\mathbf{T}_G$ is $\mathcal{P}_{\mathsf{proper}}$-free, we observe again that $zz' \in E$. It follows, by the choice of x and y that $K = \{z \in V(G) \mid x \prec_{\mathbf{T}_G} z \prec_{\mathbf{T}_G} y\}$, proving the statement.

For the converse, let us consider three vertices x, y and z such that $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$ and $xz \in E$. Then there exists a maximal clique K containing the edge xz. By assumption, the vertices of K are consecutive on a path from r. As K contains x and z, it also contains y. Thereby, we have that $xy \in E$ and $yz \in E$. It follows that $\mathbf{T}_G$ is $\mathcal{P}_{\mathsf{indifference}}$-free and so an indifference tree-layout.

$(1 \Leftrightarrow 4)$ Suppose that $\mathbf{T}_G$ is an indifference tree-layout. Let x and y be two vertices such that $x \prec_{\mathbf{T}_G} y$. Suppose that there exists z such that $z \prec_{\mathbf{T}} x$ and $zy \in E$. As $\mathbf{T}_G$ is $\{\langle \overline{12}, 13, \overline{23} \rangle, \langle \overline{12}, 13, 23 \rangle, \langle 12, 13, \overline{23} \rangle\}$-free (2), $xz \in E$, which implies that $N(y) \cap A_{\mathbf{T}}(x) \subseteq N(x) \cap A_{\mathbf{T}}(x)$. Similarly, suppose that there exists z such that $y \prec_{\mathbf{T}} z$ and $xz \in E$. As $\mathbf{T}$ is $\{\langle \overline{12}, 13, \overline{23} \rangle, \langle \overline{12}, 13, 23 \rangle, \langle 12, 13, \overline{23} \rangle\}$-free, $yz \in E$, which implies that $N(x) \cap D_{\mathbf{T}}(y) \subseteq N(y) \cap D_{\mathbf{T}}(y)$.

For the converse, let us consider three vertices x, y and z such that $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$ and $xz \in E$. As $N(z) \cap A_{\mathbf{T}_G}(y) \subseteq N(y) \cap A_{\mathbf{T}_G}(y)$, $xz \in E$ implies that $yx \in E$. Similarly, as $N(x) \cap D_{\mathbf{T}}(y) \subseteq N(y) \cap D_{\mathbf{T}}(y)$, $xz \in E$ implies that $yz \in E$. It follows that $\mathbf{T}_G$ is $\mathcal{P}_{\mathsf{indifference}}$-free and so an indifference tree-layout. $\qquad\square$

Clearly, as an indifference layout is an indifference tree-layout, every proper interval graph is a proper chordal graph. Also, as $\mathcal{P}_{\mathsf{chordal}} \subset \mathcal{P}_{\mathsf{proper}}$, proper chordal graphs are chordal graphs. The following observation proves that not every chordal graph is a proper chordal graph. For $k \geq 3$, the k-sun is a graph on $2k$ vertices $X \cup Y$ with $X = \{x_1, \dots x_k\}$ and $Y = \{y_1, \dots y_k\}$

such that X is a clique and Y is an independent set and for every $i \in [k]$, y_i is adjacent to x_i and $x_{(i+1) \mod k}$ (see Figure 5).

Observation 2. *For every, $k \geq 3$, the k-sun is not a proper chordal graph.*

Proof. We prove the claim for $k = 3$. Its generalization to arbitrary value of $k \geq 3$ is straightforward.

Let G be the 3-sun with the vertex set as in Figure 5. Suppose that $\mathbf{T}_G = (T, r, \rho_G)$ is an indifference tree-layout of G. Let b, c and d be the three non-simplicial vertices of the 3-sun G. As they form a maximal clique, these three vertices are mapped to consecutive nodes of the same path from the root to a leaf of $\mathbf{T}$ (see Theorem 6). Suppose without loss of generality that $b \prec_{\mathbf{T}_G} c \prec_{\mathbf{T}_G} d$. Let e be the simplicial vertex of G adjacent to b and d. As b, c and d are consecutive in $\mathbf{T}_G$, either $e \prec_{\mathbf{T}_G} b$ or $d \prec_{\mathbf{T}_G} e$. We observe that as e is non-adjacent to c, in both cases $\mathbf{T}_G$ is not $\mathcal{P}_{\mathsf{proper}}$-free: contradiction. $\square$

2.4 Relationship between proper chordal graphs and subclasses of chordal graphs.

Let us now discuss the relationship between proper chordal graphs and important subclasses of chordal graphs, namely interval graphs, directed path graphs and strongly chordal graphs. Figure 6 summarizes these relations.
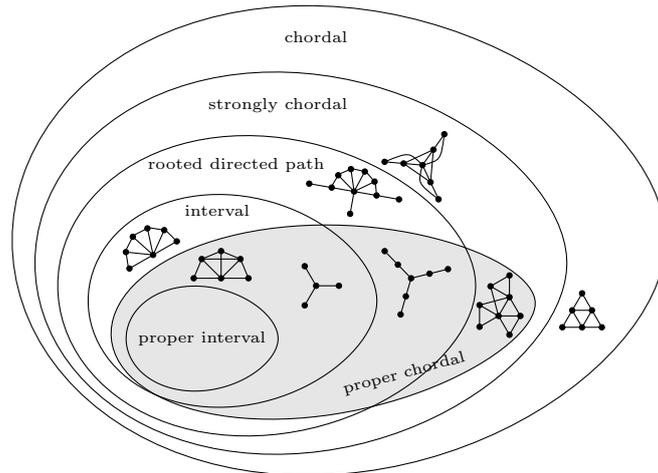
Figure 6: Relationship between proper chordal graphs and subclasses of chordal graphs.

Interval graphs. Trees are proper chordal graphs but not interval graphs. For $k \geq 3$, the k-*fan* is the graph obtained by adding a universal vertex to the path on $k+1$ vertices (see Figure 7). It is easy to see that for every $k \geq 3$, the k-fan is an interval graph. By Observation 3, we show that the 5-fan is not a proper chordal graphs. It follows that proper chordal graphs and interval graphs form incomparable graph classes.

Observation 3. *The k-fan, for $k \geq 5$, is an interval graph but not a proper chordal graph.*

Proof. First, for every k, the k-fan is an interval graph. As proper chordal graphs is an hereditary class, it suffices to prove that the 5-fan is not proper chordal. We first prove that the 4-fan

Figure 7: The 5-fan.

G has a unique indifference tree-layout $\mathbf{T}_G$. Let v be the universal vertex of G and $v_1, \ldots v_5$ be the vertices of the path appearing in the natural ordering. For $i \in [1, 4]$, we let C_i denote the maximal clique $\{v, v_i, v_{i+1}\}$. Suppose that $\mathbf{T}_G = (T, r, \rho_G)$ is an indifference tree-layout of G. By Theorem 6, every maximal clique C_i has to occur consecutively on a path of T from r to some lea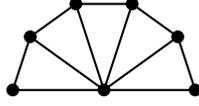f. Observe that $C_1 \cap C_2$ and $C_3 \cap C_3$ implies that $v_2 \prec_{\mathbf{T}_G} v \prec_{\mathbf{T}_G} v_3$ or that $v_3 \prec_{\mathbf{T}_G} v \prec_{\mathbf{T}_G} v_2$. Similarly, $C_2 \cap C_3$ and $C_3 \cap C_4$ implies that $v_3 \prec_{\mathbf{T}_G} v \prec_{\mathbf{T}_G} v_4$ or that $v_4 \prec_{\mathbf{T}_G} v \prec_{\mathbf{T}_G} v_3$. It follows that $v_3 \prec_{\mathbf{T}_G} v$, $v \prec_{\mathbf{T}_G} v_2$ and $v \prec_{\mathbf{T}_G} v_4$ but v_2 and v_4 are not on a common path from the root of T. In turn, this forces $v_2 \prec_{\mathbf{T}_G} v_1$ and $v_4 \prec_{\mathbf{T}_G} v_5$ and this is the unique indifference tree-layout of the 4-fan G.

Now a 5-fan H can be obtained from the 4-fan G by adding a vertex v_6 adjacent to v_5 and v, forming a new maximal clique $C_5 = \{v, v_5, v_6\}$. Clearly, H is a proper chordal graph if and only if $\mathbf{T}_G$ could be extended to an indifference tree-layout of H where C_5 is consecutive, which is not possible since in $\mathbf{T}_G$, the vertices v and v_5 are not consecutive. $\qquad\square$

Rooted directed path graphs. Rooted directed path graphs form an interesting graph class that is sandwiched between interval graphs and chordal graphs. A graph $G = (V, E)$ is a *directed path graph* if and only if it is the intersection graph of a set of directed subpaths (a subpath from the root to a leaf) of a rooted tree [38].
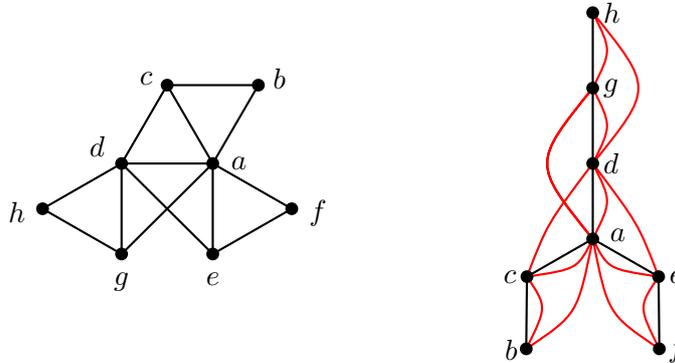
Figure 8: The Cévenol graph and an indifference tree-layout of the Cévenol graph.

Observation 4. *The Cévenol graph is a proper chordal graph but not a rooted directed path graph*

Proof. The tree-layout of Figure 8 certifies that the Cévenol graph is a proper chordal graph. To see that the Cévenol graph is not a rooted directed path graph, it suffices to observe that each of its tree-intersection models has a subtree with a branching node (either the subtree associated to a or the one associated to d). $\qquad\square$

Strongly chordal graphs. A vertex v of a graph $G = (V, E)$ is *simple* if for all $x, y \in N[v]$, $N[x] \subseteq N[y]$ or $N[y] \subseteq N[x]$. Observe that a simple vertex is a simplicial vertex. A simplicial elimination ordering $\mathbf{L}$ is a *strong perfect elimination ordering* if for every w, x, y, z such that $w \prec_{\mathbf{L}} x \prec_{\mathbf{L}} y \prec_{\mathbf{L}} z$, if $wy \in E$, $wz \in E$ and $xy \in E$ then $xz \in E$. A graph is strongly chordal if and only if it has a strong perfect elimination ordering [16]. Strongly chordal graphs are also characterized as being the k-sun free chordal graphs. As a consequence of Observation 2, we observe that every proper chordal graph is a strongly chordal graph. Moreover the inclusion between proper chordal graphs and strongly chordal is strict as the 5-fan is strongly chordal, but not proper chordal (see Observation 3).

3 FPQ-trees and FPQ-hierarchies

Let $\mathcal{P}$ be a set of patterns. In general, a graph $G \in \mathfrak{L}(\mathcal{P})$ admits several $\mathcal{P}$-free layouts. A basic example is the complete graph K_ℓ on ℓ vertices, which is a proper interval graph. It is easy to observe that every layout of K_ℓ is an indifference layout (i.e. a $\mathcal{P}_{\mathsf{proper}}$-free layout), but also a $\mathcal{P}_{\mathsf{int}}$-free layout and a $\mathcal{P}_{\mathsf{chordal}}$-free layout. Let us discuss in more details the case of proper interval graphs and interval graphs.

Proper interval graphs. Two vertices x and y of a graph G are *true-twins* if $N[x] = N[y]$. It is easy to see that the true-twin relation is an equivalence relation. If G contains some true-twins, then, by Theorem 4, the vertices of any equivalence class occurs consecutively (and in arbitrary order) in an indifference layout. It follows that for proper interval graphs, the set of indifference layouts depends on the true-twin equivalence classes. Indeed, a proper interval graph G without any pair of true-twins has a unique (up to reversal) indifference layout.

Interval graphs. In the case of interval graphs, the set of intersection models, and hence of $\mathcal{P}_{\mathsf{int}}$-free layouts, is structured by means of *modules* [18], in a similar way to the true-twin equivalence classes for $\mathcal{P}_{\mathsf{proper}}$-free layouts. A subset M of vertices of a graph G is a *module* if for every $x \notin M$, either $M \subseteq N(x)$ or $M \subseteq \overline{N}(x)$. Observe that a true-twin equivalence class is a module. A graph may have exponentially many modules. For example, every subset of vertices of the complete graph is a module. Hsu [30] proved that interval graphs having a unique intersection model are those without any trivial module.

The set of modules of a graph forms a so-called *partitive family* [9] and can thereby be represented through a linear size tree, called the *modular decomposition tree* (see [26] for a survey on modular decomposition). To recognize interval graphs in linear time, Booth and Lueker [7] introduced the concept of PQ-trees which is closely related to the modular decomposition tree or more generally to the theory of (weakly-)partitive families [9, 11]. Basically, a PQ-tree on a set X is a labelled ordered tree having X as its leaf set. Since every ordered tree defines a permutation of its leaf set, by defining an equivalence relation based on the labels of the node, every PQ-tree can be associated to a set of permutations of X. In the context of interval graphs, X is the set of maximal cliques and a PQ-tree represents the set of so-called *consecutive orderings of the maximal cliques* characterizing interval graphs.

Proper chordal graphs. As shown by Figure 9, a proper chordal graph can also have several indifference tree-layouts. In order to represent the set of $\mathcal{P}_{\mathsf{proper}}$-tree-layouts of a given graph,

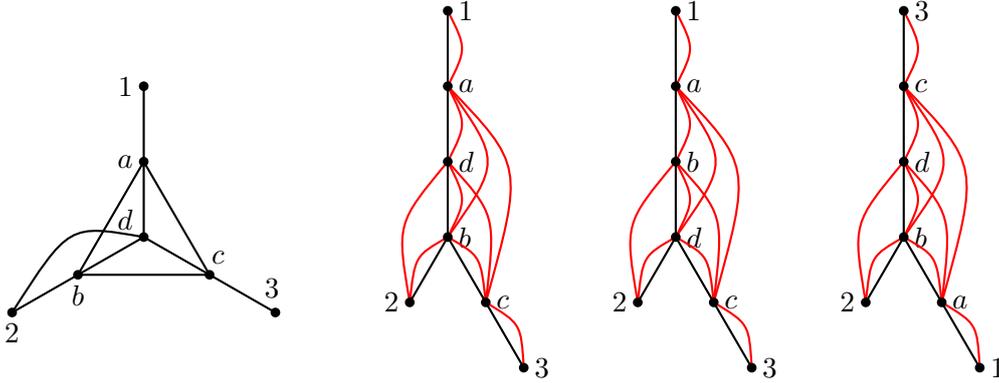we will define a structure called *FPQ-hierarchies*, based on FPQ-trees [35], a variant of PQ-trees.

Figure 9: A graph G with three possible indifference tree-layouts, two of them rooted at vertex 1, the third one at vertex 3.

3.1 FPQ-trees.

An *FPQ-tree* on the ground set X is a labelled, ordered tree T such that its leaf set $\mathcal{L}(\mathsf{T})$ is mapped to X. The internal nodes of T are of three types, F-nodes, P-nodes, and Q-nodes. If $|X| = 1$, then T is the tree defined by a leaf and a Q-node as the root. Otherwise, F-nodes and Q-nodes have at least two children while P-nodes have at least three children.

Let T and T' be two FPQ-trees. We say that T and T' are isomorphic if they are isomorphic as labelled trees. We say that T and T' are *equivalent*, denoted $\mathsf{T} \equiv_{\mathsf{FPQ}} \mathsf{T}'$, if one can be turned into a labelled tree isomorphic to the other by a series of the following two operations: $\mathsf{permute}(u)$ which permutes in any possible way the children of a P-node u; and $\mathsf{reverse}(u)$ which reverses the ordering of the children of a Q-node u. It follows that the equivalence class of an FPQ-tree T on X defines a set $\mathfrak{S}_{\mathsf{FPQ}}(\mathsf{T})$ of permutations of X.

Let $\mathfrak{S}$ be a subset of permutations of X. A subset $I \subseteq X$ is a *factor* of $\mathfrak{S}$ if in every permutation of $\mathfrak{S}$, the elements of I occur consecutively. It is well known that the set of factors of a set of permutations form a so called *weakly-partitive family* [9, 11]. As a consequence, we have the following property, which was also proved in [37].

Lemma 3. *[9, 11, 37] Let $\mathfrak{S}_{\mathsf{T}}$ be the subset of permutations of a non-empty set X associated to a PQ-tree T. Then a subset $I \subseteq X$ is a factor of $\mathfrak{S}_{\mathsf{T}}$ if and only if there exists an internal node u of T such that*

- *either $I = \mathcal{L}_{\mathsf{T}}(u)$;*

- *or u is a Q-node and there exists a set of children $v_1, \ldots, v_s$ of u that are consecutive in $<_{T,u}$ and such that $I = \bigcup_{1 \le i \le s} \mathcal{L}_{\mathsf{T}}(v_i)$.*

We observe that $u = \mathsf{lca}_{\mathsf{T}}(I)$.

Given a set $\mathcal{S} \subseteq 2^X$ of subsets of the ground set X, we let $\mathsf{Convex}(\mathcal{S})$ denote the set of permutations of X such that for every $S \in \mathcal{S}$, S is a factor of $\mathsf{Convex}(\mathcal{S})$.

Lemma 4. *Let X be a non-empty set and let $\mathcal{S} \subseteq 2^X$. In linear time in $|\mathcal{S}|$, we can compute a PQ-tree T on X such that $\mathfrak{S}_\mathsf{T} = \mathsf{Convex}(\mathcal{S})$ or decide that $\mathsf{Convex}(\mathcal{S}) = \emptyset$.*

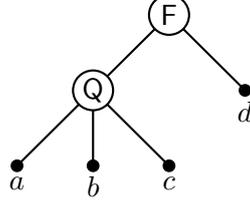*Proof.* This follows from known recognition algorithms of convex bipartite graphs (see [25]). □

Figure 10: An FPQ-tree T with $\mathfrak{S}_{\mathsf{FPQ}}(\mathsf{T}) = \{abcd, cbad\}$. The set of non-trivial common factors of $\mathfrak{S}_\mathsf{T}$ is $\mathcal{I} = \{\{a,b,c\}, \{a,b\}, \{b,c\}\}$. But observe that $\mathsf{Convex}(\mathcal{I})$ also contains the permutations $dabc$ and $dcba$ as well.

A set $\mathcal{N} \subseteq 2^X$ of subsets of X is *nested* if for every $Y, Z \in \mathcal{N}$, either $Y \subseteq Z$ or $Z \subseteq Y$. Let $\mathcal{C}$ be a collection of nested sets $\mathcal{N}_i \subseteq 2^X$ ($1 \leq i \leq k$), denoted $\mathcal{C} = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$. Observe that a subset $Y \subseteq X$ may occur in several nested sets of $\mathcal{C}$. We let $\mathcal{S} \subseteq 2^X$ denote the union of the nested sets of $\mathcal{C}$, that is $\mathcal{S} = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$. We say that a permutation $\sigma \in \mathsf{Convex}(\mathcal{S})$ is *$\mathcal{C}$-nested* if for every $1 \leq i \leq k$ and every pair of sets $Y, Z \in \mathcal{N}_i$ such that $Z \subset Y$, then $Y \setminus Z \prec_\sigma Z$. We let $\mathsf{Nested\text{-}Convex}(\mathcal{C}, \mathcal{S})$ denote the subset of permutations of $\mathsf{Convex}(\mathcal{S})$ that are $\mathcal{C}$-nested.

Lemma 5. *Let $\mathcal{C} = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ be a collection of nested sets such that for every $1 \leq i \leq k$, $\mathcal{N}_i \subset 2^X$. If $\mathsf{Nested\text{-}Convex}(\mathcal{C}, \mathcal{S}) \neq \emptyset$, with $\mathcal{S} = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$, then there exists an FPQ-tree T on X such that $\mathfrak{S}_\mathsf{T} = \mathsf{Nested\text{-}Convex}(\mathcal{C}, \mathcal{S})$. Moreover, such an FPQ-tree, when it exists, can be computed in polynomial time.*

Proof. We first prove the following claim. Let X be a non-empty set and consider $\mathcal{S} \subseteq 2^X$ containing two subsets S_1, S_2 of S such that $S_1 \subset S_2$. We set $S' = S_2 \setminus S_1$ and $\mathcal{S}' = \mathcal{S} \cup \{S'\}$.

Claim 1. $\mathsf{Convex}(\mathcal{S}') = \{\sigma \in \mathsf{Convex}(\mathcal{S}) \mid S_2 \setminus S_1 \prec_\sigma S_1 \text{ or } S_1 \prec_\sigma S_2 \setminus S_1\}$.

First observe that if $\sigma \in \mathsf{Convex}(\mathcal{S})$, satisfies $S_2 \setminus S_1 \prec_\sigma S_1$ or $S_1 \prec_\sigma S_2 \setminus S_1$, then $\sigma \in \mathsf{Convex}(\mathcal{S}')$. Now consider $\sigma' \in \mathsf{Convex}(\mathcal{S}')$. Since S_1 and $S_2 \setminus S_1$ are disjoint sets that both appear as intervals in σ', we have that either $S_1 \prec_{\sigma'} S_2 \setminus S_1$ or $S_2 \setminus S_1 \prec_{\sigma'} S_1$, proving the claim.

Let us now consider a collection $\mathcal{C} = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ of nested sets on X. Let $\mathcal{N}_i \in \mathcal{C}$ be a non trivial nested set, i.e. containing at least two subsets of X. Let $S_i^{\min}$ and $S_i^{\max}$ be respectively the largest and the smallest subset of $\mathcal{N}_i$. We set $S_i' = S_i^{\max} \setminus S_i^{\min}$. For every non-trivial nested set $\mathcal{N}_i$, we add to $\mathcal{S}$ the subset S_i' resulting in $\mathcal{S}' \in 2^X$. By the observation above, we have $\mathsf{Nested\text{-}Convex}(\mathcal{C}, \mathcal{S}) \subseteq \mathsf{Convex}(\mathcal{S}')$. By Lemma 4, we can compute (in linear time) a PQ-tree T_1 such that $\mathfrak{S}_{\mathsf{T}_1} = \mathsf{Convex}(\mathcal{S}')$.

To compute an FPQ-tree T on X such that $\mathfrak{S}_\mathsf{T} = \mathsf{Nested\text{-}Convex}(\mathcal{C}, \mathcal{S})$, the rest of the algorithm consists in freezing some Q-nodes of T_1 into F-nodes. To that aim, we first prove the following claim.

Claim 2. *If $\mathcal{N}_i \in \mathcal{C}$ is non-trivial, then in T_1, the least common ancestor u of the elements of $S_i^{\max}$ is a Q-node.*

16

Suppose towards a contradiction that u is a P-node. By Lemma 3, this implies that $\mathcal{L}_{\mathsf{T}_1}(u) = S_i^{\max}$. By definition of S_i', S_i' and $S_i^{\min}$ partition the set $S_i^{\max}$, and thereby $\mathcal{L}_{\mathsf{T}_1}(u)$. Let u_1 and u_2 be the least common ancestors of the elements of $S_i^{\min}$ and of the elements of S_i', respectively. Observe that for Lemma 3 to also holds for S_i' and $S_i^{\min}$, u_1 and u_2 have to be two distinct children of u. But as a P-node, u has at least three children, implying that $\mathcal{L}_{\mathsf{T}_1}(u_1) \cup \mathcal{L}_{\mathsf{T}_1}(u_2) \neq \mathcal{L}_{\mathsf{T}_1}(u)$: a contradiction.

Claim 2 allows to process T_1 and $\mathcal{C}$ to compute the desired FPQ-tree T as follows. Initially, we set $\mathsf{T} = \mathsf{T}_1$. For every non-trivial nested set $\mathcal{N}_i \in \mathcal{C}$, let u_i be the node that is the least common ancestor of the element of $\mathcal{S}_i^{\max}$. By Claim 2, u_i is not a P-node. If u_i is a Q-node, then we freeze it into an F-node so that for every permutation $\sigma \in \mathfrak{S}_{\mathsf{T}}$ we have that $S_i' \prec_\sigma S_i^{\min}$. Otherwise, u_i is an F-node and we check if in every permutation $\sigma \in \mathfrak{S}_{\mathsf{T}}$ we have that $S_i' \prec_\sigma S_i^{\min}$. This can be done easily in polynomial time. $\qquad\square$

3.2   FPQ-hierarchies.

A *hierarchy of ordered trees* H is defined on a set $\mathcal{T} = \{T_0, T_1, \ldots, T_p\}$ of non-trivial ordered trees arranged in an edge-labelled tree, called the *skeleton tree* S_{H}. More formally, for $0 < i \leq p$, the root r_i of T_i is attached, through a *skeleton edge* e_i, to an internal node f_i of some tree T_j with $j < i$. Suppose that $e_i = r_i f_i$ is the sketelon edge linking the root r_i of T_i to a node f_i of T_j having c children. Then the label of e_i is a pair of integers $I(e_i) = (a_i, b_i) \in [c] \times [c]$ with $a_i \leq b_i$. The contraction of the trees of $\mathcal{T}$ in a single node each, results in the skeleton tree S_{H}.

From a hierarchy of ordered trees H, we define a rooted tree T_{H} whose node set is $\bigcup_{0 \leq i \leq p} \mathcal{L}(T_i)$ and that is built as follows. The root of T_{H} is ℓ_0 the first leaf of $\mathcal{L}(T_0)$ in σ_{T_0}. For every $0 \leq i \leq p$, the permutation σ_{T_i} of $\mathcal{L}(T_i)$, defined by T_i, is a path of T_{H}. Finally, for $1 \leq i \leq p$, let ℓ_i be the first leaf of $\mathcal{L}(T_i)$ in σ_{T_i}. Suppose T_i is connected in H to T_j through the skeleton edge $e_i = r_i f_i$ with label $I(e_i) = (a_i, b_i)$. Let u_j the b_i-th child of f_i in T_j and ℓ be the leaf of T_j that is a descendant of u_j and largest in σ_{T_j}. Then set ℓ as the parent of ℓ_i in T_{H}. See Figure 11 for an example.

An *FPQ-hierarchy* is a hierarchy of FPQ-trees with an additional constraint on the labels of the skeleton edges. Let $e_i = r_i f_i$ be the skeleton from the root r_i of T_i to the node f_i of T_j with $j \leq i$. If f_j is a P-node with c children, then $I(e_i) = (1, c)$. As in the case of FPQ-trees, we say two FPQ-hierarchies H and H' are isomorphic if they are isomorphic as labeled ordered trees. That is the types of the nodes, the skeleton edges and their labels are preserved. We say that H and H' are equivalent, denoted $\mathsf{H} \approx_{\mathsf{FPQ}} \mathsf{H}'$, if one can be turned into an FPQ-hierarchy isomorphic to the other by a series of permute(u) and reverse(u) operations (with u being respectively a P-node and a Q-node) to modify relative ordering of the tree-children of u. Suppose that u is a Q-node with c children incident to a skeleton edge e. Then applying reverse(u) transforms $I(e) = (a, b)$ into the new label $I^c(e) = (c + 1 - b, c + 1 - a)$. It follows that the equivalence class of an FPQ-hierarchy H on the set $\mathcal{T} = \{\mathsf{T}_0, \mathsf{T}_1, \ldots, \mathsf{T}_p\}$ of FPQ-trees defines a set $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H})$ of rooted trees on $\bigcup_{0 \leq i \leq p} \mathcal{L}(T_i)$. Observe that since reversing a Q-node modifies the labels of the incident skeleton edges, two rooted trees of $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H})$ may not be isomorphic (see Figure 11).
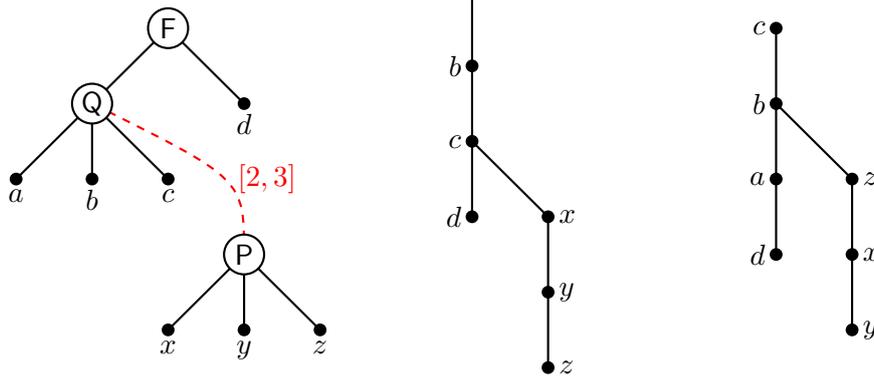
Figure 11: An FPQ-hierarchy H. The set $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H})$ contains 12 rooted trees, two of which are depicted. Observe that from the left to the right tree, the ordering on the leaves of the Q-node is reversed and that the ordering on the leaves of the P-nodes are different. In both trees however, the path containing $\{x, y, z\}$ is attached below the leaves $\{b, c\}$ since these leaves form the interval $[2, 3]$ of the Q-node and this interval is the label of the unique skeleton edge.

4 Compact representation of the set of indifference tree-layouts

In this section, we show how, for a given proper chordal graph G, an FPQ-hierarchy H can be constructed to represent for a given vertex $x \in V$ the set of indifference tree-layout rooted at a vertex x (if such an indifference tree-layout exists). To that aim, we first provide a characterization of indifference tree-layout alternative to Theorem 6. This characterization naturally leads us to define the notion of *block* that for a fixed vertex x of a proper chordal graph, drives the structure and the combinatorics of the set of indifference tree-layouts rooted at x.

4.1 Blocks and indifference tree-layouts

Let S be a non-empty vertex subset of a connected graph $G = (V, E)$ and let C be a connected component of $G - S$. We say that $x \in C$ is *S-maximal* if for every vertex $y \in C$, $N(y) \cap S \subseteq N(x) \cap S$. Observe that if C contains two distinct S-maximal vertices x and y, then $N(x) \cap S = N(y) \cap S$.

Definition 3. *Let S be a subset of vertices of a graph $G = (V, E)$ and let C be a connected component of $G - S$. A maximal subset of vertices $X \subseteq C$ is an S-block, if every vertex of X is S-maximal and $(N(S) \cap C)$-universal.*

We let the reader observe that a connected component C of $G - S$ may not contain an S-block. However, if C contains an S-block, then it is uniquely defined.

Lemma 6. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be a tree-layout of a graph $G = (V, E)$. Then $\mathbf{T}_G$ is an indifference tree-layout if and only if for every vertex x distinct from $\rho_G^{-1}(r)$, x is $A_{\mathbf{T}_G}(x)$-maximal and $(N(A_{\mathbf{T}_G}(x)) \cap D_{\mathbf{T}_G}[x])$-universal.*

Proof. Suppose that every vertex x distinct from $\rho_G^{-1}(r)$, is $A_{\mathbf{T}_G}(x)$-maximal and $A_{\mathbf{T}_G}(x)$-universal. Let y, z be two distinct vertices of G such that $y \prec_{\mathbf{T}_G} x \prec_{\mathbf{T}_G} z$ and $yz \in E$. Observe

18

that $y \in A_{\mathbf{T}_G}(x)$ and $z \in N(A_{\mathbf{T}_G}(x))$. As x is $A_{\mathbf{T}_G}(x)$-maximal and $(N(A_{\mathbf{T}_G}(x)) \cap D_{\mathbf{T}_G}[x])$-universal, $yz \in E$ implies that $xy \in E$ and $xz \in E$. So $\mathbf{T}_G$ is an indifference tree-layout.

To prove the reverse, consider $x \in V$ distinct from $\rho_G^{-1}(r)$ and C the connected component of $G - A_{\mathbf{T}_G}(x)$ containing x. First observe that as $\mathbf{T}_G$ is a tree-layout of G, every vertex $z \in C$ is a descendant of x is $\mathbf{T}_G$. Suppose that either x is not $A_{\mathbf{T}_G}(x)$-maximal or not $(N(A_{\mathbf{T}_G}(x)) \cap D_{\mathbf{T}_G}[x])$-universal. In the former case, there exist two vertices $y \in A_{\mathbf{T}_G}(x)$ and $z \in C$ such that $yz \in E$ but $xy \notin E$. In the latter case, there exists $y \in A_{\mathbf{T}}(x)$ and $z \in C$ such that $yz \in E$ but $xy \notin E$. By the above observation, we have that $z \prec_{\mathbf{T}_G} x \prec_{\mathbf{T}_G} y$ and so in both cases $\mathbf{T}_G$ is not an indifference tree-layout. $\square$

So Lemma 6 and the discussion above shows that if $\mathbf{T}_G = (T, r, \rho_G)$ is an indifference tree-layout of a proper chordal graph $G = (V, E)$, then every vertex x distinct from $\rho_G^{-1}(r)$ can be associated with a non-empty $A_{\mathbf{T}_G}(x)$-block which contains x. Hereafter, we let $B_{\mathbf{T}_G}(x)$ denote the $A_{\mathbf{T}_G}(x)$-block containing x. If $x = \rho_G^{-1}(r)$, then we set $B_{\mathbf{T}_G}(x) = \{x\}$.

Lemma 7. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of a connected proper chordal graph $G = (V, E)$. For every vertex $x \in V$, the vertices of the block $B_{\mathbf{T}_G}(x)$ appear consecutively on a path rooted at x and induces a clique in G.*

Proof. The property trivially holds for $x = \rho_G^{-1}(r)$. Let us consider $x \neq \rho_G^{-1}(r)$ and let C be the connected component of $G - A_{\mathbf{T}_G}(x)$ containing x. By the definition of blocks, every vertex of $B_{\mathbf{T}_G}(x)$ is $(N(A_{\mathbf{T}_G}(x)) \cap D_{\mathbf{T}_G}[x])$-universal. This implies that $B_{\mathbf{T}_G}(x)$ induces a clique. Observe that in every tree-layout, the vertices of a clique appear on a path from the root to a leaf. By definition of $B_{\mathbf{T}_G}(x)$, every vertex of $B_{\mathbf{T}_G}(x)$ distinct from x is a descendant of x in $\mathbf{T}_G$. Let z be the lowest vertex of $B_{\mathbf{T}_G}(x)$ in $\mathbf{T}_G$ and let $y \in V$ be a vertex such that $x \prec_{\mathbf{T}_G} y \prec_{\mathbf{T}_G} z$. Then observe that by the indifference property of $\mathbf{T}_G$, y is $A_{\mathbf{T}_G}(x)$-maximal and y is adjacent to every vertex of $N(A_{T_G}(x))$ that is a descendant of x. It follows that $y \in B_{\mathbf{T}_G}(x)$. $\square$

Lemma 8. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of a connected proper chordal graph $G = (V, E)$. If $y \in B_{\mathbf{T}_G}(x)$, then $B_{\mathbf{T}_G}(y) \subset B_{\mathbf{T}_G}(x)$.*

Proof. Suppose that there exists $z \in B_{\mathbf{T}_G}(y) \setminus B_{\mathbf{T}_G}(x)$. Let us observe that x, y and z are pairwise adjacent: $xy \in E$ because $y \in B_{\mathbf{T}_G}(x)$; $yz \in E$ because $z \in B_{\mathbf{T}_G}(y)$; and finally, $xz \in E$ because $x \in A_{\mathbf{T}_G}(y)$ and $N(z) \cap A_{\mathbf{T}_G}(y) = N(y) \cap A_{\mathbf{T}_G}(y)$. As $A_{\mathbf{T}_G}(x) \subset A_{\mathbf{T}_G}(y)$, we also have that $N(x) \cap A_{\mathbf{T}_G}(x) = N(y) \cap A_{\mathbf{T}_G}(x) = N(z) \cap A_{\mathbf{T}_G}(x)$. So the fact that $z \notin B_{\mathbf{T}_G}(x)$, implies that there exists $v \in N(A_{\mathbf{T}_G}(x)) \cap (D_{\mathbf{T}_G}(x) \cup \{x\})$ such that $vz \notin E$. As $xz \in E$, we know that $x \neq v$. As $y \in B_{\mathbf{T}_G}(x)$, we also have $yv \in E$. It follows that $v \in N(A_{\mathbf{T}_G}(y)) \cap (D_{\mathbf{T}_G}(y) \cup \{y\})$. Finally, since $z \in B_{\mathbf{T}_G}(y)$, we also have $vz \in E$: contradiction. $\square$

Lemma 9. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of a connected proper chordal graph $G = (V, E)$. For every vertex x, $G[D_{\mathbf{T}_G}(x)]$ is connected.*

Proof. Let x and y be two vertices such that x is the parent of y in $\mathbf{T}_G$. We prove that $xy \in E$. For the sake of contradiction, suppose that $xy \notin E$. Observe that $N(x) \cap D_{\mathbf{T}_G}(x) = \emptyset$, as otherwise $\mathbf{T}$ would not be an indifference tree-layout. For the same reason, we have that for every $z \in D_{\mathbf{T}}(y)$, $N(z) \cap A_{\mathbf{T}_G}(y) = \emptyset$. This implies that $D_{\mathbf{T}_G}(x)$ and $A_{\mathbf{T}_G}(y)$ are not in the same connected component of G: contradiction. $\square$

4.2 The block tree of an indifference tree-layout

Given an indifference tree-layout $\mathbf{T}_G = (T, r, \rho)$ of a connected proper chordal graph $G = (V, E)$, we define the set $\mathcal{B}(\mathbf{T}_G)$ as the set containing the inclusion-maximal blocks of $\mathbf{T}_G$. Observe that $\{\rho^{-1}(r)\} \in \mathcal{B}(\mathbf{T}_G)$. Corollary 1 is a direct consequence of Lemma 8.

Corollary 1. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of a connected proper chordal graph $G = (V, E)$. Then, $\mathcal{B}(\mathbf{T}_G)$ is a partition of V.*

From Lemma 7 and Corollary 1, we can define the *block tree* of $\mathbf{T}_G$, which will be denoted $\mathbf{T}_G|_{\mathcal{B}_{\mathbf{T}(G)}}$, obtained by contracting every block of $\mathcal{B}(\mathbf{T}_G)$ into a single node.

Lemma 10. *Let $\mathbf{T}_G = (T, r, \rho_G)$ and $\mathbf{T}'_G = (T', r', \rho'_G)$ be two indifference tree-layouts of a connected proper chordal graph $G = (V, E)$ both having the same root $x = \rho_G^{-1}(r) = \rho_G'^{-1}(r')$. Then $\mathcal{B}(\mathbf{T}_G) = \mathcal{B}(\mathbf{T}'_G)$ and $T_G|_{\mathcal{B}(\mathbf{T}_G)}$ and $T'_G|_{\mathcal{B}(\mathbf{T}'_G)}$ are isomorphic trees.*

Proof. We prove the property by induction on the inclusion-maximal blocks of $\mathbf{T}_G$ and $\mathbf{T}'_G$. We have that $\{x\}$ belongs to both $\mathcal{B}(\mathbf{T}_G)$ and $\mathcal{B}(\mathbf{T}'_G)$. Let us assume that the property holds for a subset $\mathcal{B} \subset \mathcal{B}(\mathbf{T}_G) \cap \mathcal{B}(\mathbf{T}'_G)$ of maximal blocks inducing a connected subtree of $\mathbf{T}_G$ and $\mathbf{T}'_G$ containing the root block $\{x\}$. Let S be the set of vertices belonging to blocks in $\mathcal{B}$. Let C be a connected component of $G - S$. By Lemma 9, C induces a subtree of $\mathbf{T}_G$ (respectively of $\mathbf{T}'_G$) such that the root y (respectively y') of that subtree is the child of some vertex z of some block in $\mathcal{B}$ (respectively z'). Now observe that as $\mathbf{T}_G$ and $\mathbf{T}'_G$ are two indifference tree-layouts of G, both $B_{\mathbf{T}_G}(y)$ and $B_{\mathbf{T}'_G}(y')$ contain the vertices of C that are S-maximal vertices and that are $N(S) \cap C$-universal. It follows that $B_{\mathbf{T}_G}(y) = B_{\mathbf{T}'_G}(y')$. Set $B = B_{\mathbf{T}_G}(y)$. Moreover by the indifference property, observe that $N(B) \cap S$ is consecutive in $[r, y]_{\mathbf{T}_G}$ and contains z. Similarly, $N(B) \cap S$ is consecutive in $[r, y']_{\mathbf{T}'_G}$ and contains z'. As the induction hypothesis holds on $\mathcal{B}$, y' and z' belong to the same block of $\mathcal{B}$. It follows that the property is satisfied on $\mathcal{B} \cup \{B\}$ as well, implying that $\mathcal{B}(\mathbf{T}_G) = \mathcal{B}(\mathbf{T}'_G)$ and $T_G|_{\mathcal{B}(\mathbf{T}_G)}$ and $T'_G|_{\mathcal{B}(\mathbf{T}'_G)}$ are isomorphic trees. $\qquad\square$

It follows that the block tree only depends on the root vertex and not on a given indifference tree-layout rooted at that vertex. Thereby, from now on, we let $\mathcal{B}_G(x)$ and $\mathsf{B}_G^{\mathsf{tree}}(x)$ respectively denote the set of maximal blocks and the block tree of any indifference tree-layout of G rooted at vertex x. However, we observe that the block tree does not fully describe the possible set of indifference tree-layouts. The block tree does not reflect how each block is precisely attached to its parent block.

Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree layout of a graph $G = (V, E)$. Let $B_{\mathbf{T}_G}(x) \in \mathcal{B}(\mathbf{T}_G)$ with x distinct from the root of $\mathbf{T}_G$. We denote by $C_{\mathbf{T}_G}(x)$ the connected component of $G - A_{\mathbf{T}_G}(x)$ containing x. Let $C_1, \ldots C_k$ be the connected components of $G[C_{\mathbf{T}_G}(x) \setminus B_{\mathbf{T}_G}(x)]$. We define $\mathcal{C}_{\mathbf{T}_G}(x) = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ a collection of sets of $2^{B_{\mathbf{T}_G}(x)}$: for every $1 \leq i \leq k$ and every vertex $y \in C_i$ such that $N(y) \cap B_{\mathbf{T}_G}(x) \neq \emptyset$, then we add $N(y) \cap B_{\mathbf{T}_G}(x)$ to $\mathcal{N}_i$. We define $\mathcal{S}_x = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$.

Lemma 11. *Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of a connected proper chordal graph G. Let x be a vertex of G distinct from the root $\rho_G(r)$ such that $B_{\mathbf{T}_G}(x) \in \mathcal{B}_{\mathbf{T}_G}(G)$. Then $\mathcal{C}_{\mathbf{T}_G}(x) = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ is a collection of nested sets such that $\mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_G}(x), \mathcal{S}_x) \neq \emptyset$.*

Proof. We first argue that $\mathcal{C}_{\mathbf{T}_G}(x)$ is a collection of nested sets. Consider two vertices y and z of some connected component C_i of $G[C_{\mathbf{T}_G}(x) \setminus B_{\mathbf{T}_G}(x)]$ that both have a neighbor in $B_{\mathbf{T}_G}(x)$.

Observe that $\mathsf{lca}_{\mathbf{T}_G}(y,z)$ is a node mapped to a vertex of C_i. By the indifference property of $\mathbf{T}_G$, it follows that $N(y) \cap B_{\mathbf{T}_G}(x)$ and $N(z) \cap B_{\mathbf{T}_G}(x)$ form in $\mathbf{T}_G$ two subpaths sharing the same least boundary, implying that $\mathcal{N}_i$ is a nested set. Finally, observe the permutation σ of $B_{\mathbf{T}_G}(x)$ such that for every two vertices $y, z \in B_{\mathbf{T}_G}(x)$, $y \prec_\sigma z$ if and only if $y \in A_{\mathbf{T}_G}(z)$ is a permutation of $\mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_G}(x), \mathcal{S}_x)$. $\qquad\square$

4.3 A canonical FPQ hierarchy

We can now prove, using Lemma 3, Lemma 5, Corollary 1 and Lemma 11, the existence of a canonical FPQ-hierarchy $\mathsf{H}_G(x)$ that encodes the set of indifference tree-layouts of G rooted at x, if there exists one (see Figure 12).

Theorem 7. *Let $G = (V, E)$ be a proper chordal graph. If G has an indifference tree-layout rooted at some vertex x, then there exists an FPQ-hierarchy $\mathsf{H}_G(x)$ such that a tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ of G is an indifference tree-layout such that $\rho_G^{-1}(r) = x$ if and only if $\mathbf{T}_G \in \mathfrak{T}_{FPQ}(\mathsf{H}_G(x))$.*

Moreover $\mathsf{H}_G(x)$ is unique and, given an indifference tree-layout rooted at x, it can be computed in polynomial time.

Proof. Let $\mathbf{T}_G = (T, r, \rho_G)$ be an indifference tree-layout of G such that $\rho_G^{-1}(r) = x$. We can construct an FPQ-hierarchy $\mathsf{H}_G(x)$ as follows (see Figure 12):

1. The skeleton tree of $\mathsf{H}_G(x)$ is the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$.

2. For each block $B_{\mathbf{T}_G}(y) \in \mathcal{B}_{\mathbf{T}_G}(G)$, with $y \neq x$, compute the FPQ-tree T_y of the collection $\mathcal{C}_{\mathbf{T}_G}(y)$ of nested sets such that $\sigma_{\mathsf{T}_y} = \mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_G}(y), \mathcal{S}_y)$.

3. Suppose that $B_{\mathbf{T}_G}(z)$ is a child of $B_{\mathbf{T}_G}(y)$ in $\mathsf{B}_G^{\mathsf{tree}}(x)$. Then the parent in $\mathsf{H}_G(x)$ of the root r_z of T_z is the node $u_z = \mathsf{lca}_{\mathsf{T}_y}(I_z)$ where $I_z = N(z) \cap B_{\mathbf{T}_G}(y) \in \mathcal{S}_y$. Suppose that u has k children $v_1, \ldots, v_k$. We set the label of the skeleton-edge $r_z u$ to the interval $[h, j]$ such that $I_z = \bigcup_{h \leq i \leq j} \mathcal{L}_{\mathsf{T}_y}(v_i)$.

Claim 3. *The FPQ-hierarchy $\mathsf{H}_G(x)$ is uniquely defined and can be computed in polynomial time.*

By Lemma 10, the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$ is unique. Moreover, given $\mathbf{T}_G$, it can be computed in polynomial time. Let y be a vertex distinct from x such that $B_{\mathbf{T}_G}(y) \in \mathcal{B}_{\mathbf{T}_G}(G)$. By Lemma 11, the collection $\mathcal{C}_{\mathbf{T}_G}(y)$ verifies that $\mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_G}(y), \mathcal{S}_y) \neq \emptyset$. So by Lemma 5, the FPQ-tree T_y exists and can be computed in polynomial time. Finally, suppose that $B_{\mathbf{T}_G}(z)$ is a child of $B_{\mathbf{T}_G}(y)$ in the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$. By Lemma 3, the node u_z is well defined and the interval I_z is unique.

It directly follows from the construction above and the definition of $\mathfrak{T}_{FPQ}(\mathsf{H}_G(x))$ that $\mathbf{T}_G \in \mathfrak{T}_{FPQ}(\mathsf{H}_G(x))$.

Claim 4. *If $\mathbf{T}_G \in \mathfrak{T}_{FPQ}(\mathsf{H}_G(x))$ then $\mathbf{T}_G$ is an indifference tree-layout of G rooted at x.*

Let us argue that $\mathbf{T}_G$ is a tree-layout of G. Let y and z be two vertices of G such that neither $y \prec_{\mathbf{T}_G} z$ nor $z \prec_{\mathbf{T}_G} y$. We denote B_y and B_z the blocks of $\mathcal{B}_{\mathbf{T}_G}(G)$ respectively containing y and z. There are two cases to consider. Suppose first that B_y is an ancestor of B_z in the block tree
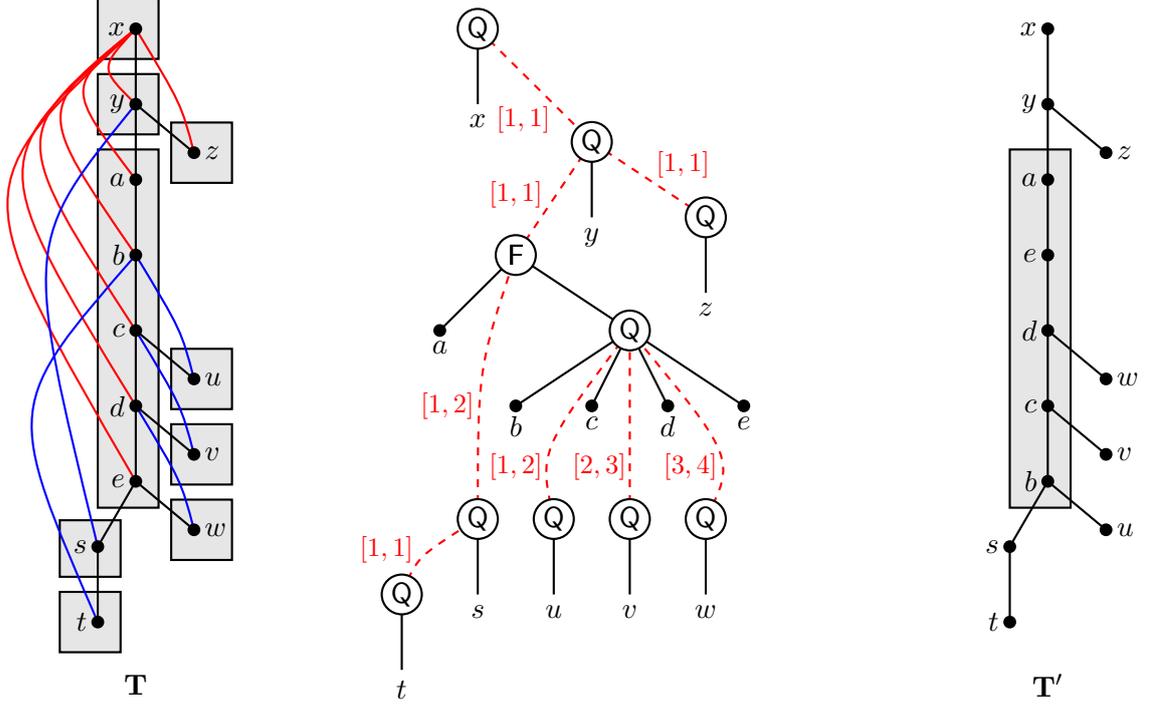
Figure 12: On the left hand side, an indifference tree-layout $\mathbf{T}$ rooted at vertex x of a proper chordal graph G. For every vertex, only the edge (blue or red) to its highest neighbor in $\mathbf{T}$ is depicted. The boxes represent the partition into blocks. The FPQ-hierarchy $\mathsf{H}_G(x)$ is depicted in the middle. Observe that for the FPQ-tree T of the block $B_{\mathbf{T}}(a) = \{a, b, c, d, e\}$, $\mathfrak{S}(\mathsf{T}) = \{abcde, aedcb\}$. It follows that $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H}_G(x))$ contains two indifference tree-layouts $\mathbf{T}$ and $\mathbf{T}'$. The latter one is shown on the right hand side.

$\mathsf{B}_G^{\mathsf{tree}}(x)$. Observe that by the choice of the permutation σ_{T_y} of B_y in the construction of $\mathbf{T}_G$, for every vertex $v \in N(z) \cap B_y$, $v \prec_{\mathbf{T}_G} \mathsf{lca}_{\mathbf{T}_G}(y, z)$. It follows that $yz \notin E$. So let us assume that B_y and B_z are not an ancestor of one another in the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$. Let $B_{\mathbf{T}_G}(u) \in \mathcal{B}_{\mathbf{T}_G}(G)$ be their least common ancestor. Observe that $B_{\mathbf{T}_G}(u) \cup A_{\mathbf{T}_G}(u)$ is a separator in G for y and z. It follows that $yz \notin E$. Thereby, $\mathbf{T}_G$ is indeed a tree-layout of G.

To prove that $\mathbf{T}_G$ is an indifference tree-layout, let us consider three distinct vertices v, y and z of V such that $y \prec_{\mathbf{T}_G} v \prec_{\mathbf{T}_G} z$ and $yz \in E$. Let $B_z \in \mathcal{B}_{\mathbf{T}_G}(G)$ be the block containing z. Suppose that $v \in B_z$. As, by Lemma 6, every block of $\mathcal{B}_{\mathbf{T}_G}(G)$ induces a clique, $vz \in E$. If $y \in B_z$, then, for the same reason, $yv \in E$. Otherwise, by definition of a block, $N(v) \cap B = N(z) \cap B$ for every block B that is an ancestor of B_y in $\mathsf{B}_G^{\mathsf{tree}}(x)$, implying that $yv \in E$. Suppose that $v \notin B_z$. Let $B_y \in \mathcal{B}_{\mathbf{T}_G}(G)$ be the block containing y. Suppose that $v \in B_y$. As B_y induces a clique, $vy \in E$. Moreover, by construction of $\mathbf{T}_G$ and the choice of $\sigma_{\mathsf{T}_y} \in \mathfrak{S}(\mathsf{T}_y)$, where T_y is the FPQ-tree associated to the block B_y, $N(z) \cap B_y$ is an interval that contains $D_{\mathbf{T}_G}(y) \cap B_y$. It follows that $zv \in E$. So assume that $v \notin B_y$. Observe then that by the definition of blocks, we have that $N(z) \cap B_y \subseteq N(v) \cap B_y$ and that y is adjacent to every vertex of B_z that has a neighbor in B_y, implying that $vy \in E$ and $vz \in E$. $\qquad\square$

5 Algorithmic aspects

This section is dedicated to the design of polynomial time algorithms for proper chordal graphs. We first tackle the recognition problem of proper chordal graphs. Then we show that using the FPQ-hierarchies, we can resolve the graph isomorphism problem between two proper chordal graphs in polynomial time. In a way, this generalizes the PQ-tree based graph isomorphism algorithm for interval graphs [37]. This latter result is of interest as it shows that proper chordal graphs extends the known limit of tractability for the graph isomorphism problem. Indeed, it is known that graph isomorphism is GI-complete on strongly chordal graphs [47].

5.1 Recognition

Given a graph $G = (V, E)$, the recognition algorithm test for every vertex $x \in V$, if G has an indifference tree-layout rooted at x. So from now on, we assume that we are given as input the graph G and a fixed vertex x. The algorithm is then two-step. First, the first step of the algorithm aims at computing the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$ of G rooted at x that would correspond to the skeleton tree of the FPQ-hierarchy $\mathsf{H}_G(x)$ if G has an indifference tree-layout rooted at x. Then, in the second step, instead of computing $\mathsf{H}_G(x)$, we verify that $\mathsf{B}_G^{\mathsf{tree}}(x)$ can indeed be turned into an indifference tree-layout of G. If eventually we can construct an indifference tree-layout, then G is proper chordal. If G is proper chordal and has an indifference tree-layout rooted at x, then the algorithm will succeed.

Computing the blocks and the block tree. Let us assume that the input graph $G = (V, E)$ is proper chordal and let us consider a vertex $x \in V$ such that x is the root of some indifference tree-layout of G. As discussed above, the first step aims at computing the skeleton tree of $\mathsf{H}_G(x)$ (see Theorem 7). That skeleton tree is the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$ that can be obtained from any indifference tree-layout rooted at x by contracting the blocks of $\mathcal{B}_G(x)$ into a single node each (see Lemma 10). To compute $\mathsf{B}_G^{\mathsf{tree}}(x)$, we perform a search on G starting at x (see Algorithm 1 below). At every step of the search, if the set of searched vertices is S, then the algorithm either identifies, in some connected component C of $G - S$, a new block S-block of $\mathcal{B}_G(x)$ and connects it to the current block tree, or (if C does not contain an S-block) stops and declares that there is no block tree rooted at x.

Lemma 12. *Let x be a vertex of a graph $G = (V, E)$. If G is proper chordal and has an indifference tree-layout rooted at x, then Algorithm 1 returns the block tree $\mathsf{B}_G^{\mathsf{tree}}(x)$ that is the skeleton tree of the FPQ-hierarchy $\mathsf{H}_G(x)$.*

Proof. Suppose that G is proper chordal and has an indifference tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ rooted at x. From Lemma 10, $\mathsf{B}_G^{\mathsf{tree}}(x)$ is well-defined and unique. To prove the correctness of Algorithm 1, we establish the following invariant. At every step, if S is the set of searched vertices, then

(1). $\mathcal{B} \subseteq \mathcal{B}_G(x)$ and $\mathsf{B}^{\mathsf{tree}} = \mathsf{B}_{G[S]}^{\mathsf{tree}}(x)$ is the unique block-tree rooted at x of $G[S]$;

(2). if C is a connected component of $G - S$, then for every pair of block, $B, B' \in \mathcal{B}$ such that $N(C) \cap B \neq \emptyset$ and $N(C) \cap B' \neq \emptyset$, either B is an ancestor of B' in $\mathsf{B}^{\mathsf{tree}}$ or vice-versa.

Algorithm 1: Block tree computation

Input: A graph $G = (V, E)$ and a vertex $x \in V$.
Output: The block tree $\mathsf{B}^{\mathsf{tree}}_G(x)$, if G has an indifference tree-layout rooted at x.

---
1   set $S \leftarrow \{x\}$, $\mathcal{B} \leftarrow \big\{\{x\}\big\}$ and $\mathsf{B}^{\mathsf{tree}} \leftarrow (\mathcal{B}, \emptyset)$;
2   **while** $S \neq V$ **do**
3   let C be a connected component of $G - S$;
4   **if** C *contains a S-block X* **then**
5         $S \leftarrow S \cup X$ and $\mathcal{B} \leftarrow \mathcal{B} \cup \big\{X\big\}$;
6   let $B \in \mathcal{B}$ such that $N(X) \cap B \neq \emptyset$ and that is the deepest;
7   add an edge in $\mathsf{B}^{\mathsf{tree}}$ between B and X;
8   **else**
9   **stop** and **return** G *has no indifference tree-layout rooted at x*;
10   **end**
11   **end**
12   **return** $\mathsf{B}^{\mathsf{tree}}$;

The invariant clearly holds when $S = \{x\}$. Let C be a connected component of $G - S$. Let $y \in C$ that is the closest to the root of $\mathbf{T}_G$. Observe that as (1) holds on $G[S]$, $X = B_{\mathbf{T}_G}(y)$ is an S-block and it belongs to $\mathcal{B}_G(x)$. By definition of $\mathsf{B}^{\mathsf{tree}}_G(x)$, observe that the parent of every block B' of $\mathcal{B}_G(x)$ is the deepest block B in $\mathsf{B}^{\mathsf{tree}}_G(x)$ such that $N(B') \cap B \neq \emptyset$. By (2), $\mathsf{B}^{\mathsf{tree}}$ contains a unique deepest block $B \in \mathcal{B}$ such that $N(C) \cap B \neq \emptyset$. It follows that if we attach X to B in $\mathsf{B}^{\mathsf{tree}}$, we extend $\mathsf{B}^{\mathsf{tree}}$ to $\mathsf{B}^{\mathsf{tree}}_{G[S \cup X]}(x)$, the block tree of $G[S \cup X]$. So (1) is still valid after adding X to the set of visited vertices. Moreover, observe that when removing X from C every connected component of $G[C] - X$ has neighbors in X, implying that (2) is maintained as well.

Finally, we observe that every step of the algorithm can be performed in polynomial time, including testing the existence of an S-block in C. $\qquad\square$

Before describing the second step of the algorithm, let us discuss some properties of $\mathsf{B}^{\mathsf{tree}}$ returned by Algorithm 1 when $\mathcal{B}$ is a partition of V. An *extension* of $\mathsf{B}^{\mathsf{tree}}$ is any tree $T_{\mathsf{B}^{\mathsf{tree}}}$ obtained by substituting every node $B \in \mathcal{B}$ by an arbitrary permutation σ_B of the vertices of B. In this construction, if B is the parent of B' in $\mathsf{B}^{\mathsf{tree}}$, x is the last vertex of B in σ_B that has a neighbor in B' and x' is the first vertex of B' in $\sigma_{B'}$, then the parent of x' in $T_{\mathsf{B}^{\mathsf{tree}}}$ is a vertex of B that appears after x in σ.

Observation 5. *Let x be a vertex of a graph $G = (V, E)$. If $\mathsf{B}^{\mathsf{tree}}$ is returned by Algorithm 1, then every extension $T_{\mathsf{B}^{\mathsf{tree}}}$ of $\mathsf{B}^{\mathsf{tree}}$ is a tree-layout of G. And moreover, if B, B', and B'' are three blocks of $\mathcal{B}$ such that $B \prec_{\mathsf{B}^{\mathsf{tree}}} B' \prec_{\mathsf{B}^{\mathsf{tree}}} B''$, then for every $y \in B$, $y' \in B'$, $y'' \in B''$, $yy'' \in E$ implies that $y'y \in E$ and $y'y'' \in E$.*

Proof. The fact that $T_{\mathsf{B}^{\mathsf{tree}}}$ is a tree-layout of G directly follows from property (2) of the invariant of Algorithm 1. Concerning triples of vertices such as y, y' and y'', observe first that $y \prec_{T_{\mathsf{B}}^{\mathsf{tree}}} y' \prec_{T_{\mathsf{B}}^{\mathsf{tree}}} y''$. The fact that $yy'' \in E$ implies that $y'y \in E$ and $y'y'' \in E$ follows from Definition 3 of a S-block and the fact that at the step B' is selected, then B'' is contained in the same connected component as B'. $\qquad\square$

Nested sets. From Observation 5, an extension of B^{tree} is not yet an indifference tree-layout of G. However, if G is a proper chordal graph that has an indifference tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ rooted at x, then, by Lemma 12, $\mathsf{B}^{\text{tree}} = \mathsf{B}_G^{\text{tree}}(x)$. It then follows from the proof of Theorem 7 that $\mathbf{T}_G$ is an extension of B^{tree}. The second step of the algorithm consists in testing if B^{tree} has an extension that is an indifference tree-layout.

By Lemma 12, we can assume that Algorithm 1 has returned $\mathcal{B}_G(x)$ and $\mathsf{B}_G^{\text{tree}}(x)$. To every block B of $\mathcal{B}_G(x)$, we assign a collection of nested subsets of 2^B which we denote $\mathcal{C}_B = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ (see Algorithm 2 for a definition of $\mathcal{C}_B$). The task of the second step of the recognition algorithm is to verify that for every block B, $\mathsf{Nested\text{-}Convex}(\mathcal{C}_B, \mathcal{S}_B) \neq \emptyset$ where $\mathcal{S}_B = \cup_{1 \leq i \leq k} \mathcal{N}_i$. This amounts to testing whether every block of $\mathsf{B}_G^{\text{tree}}(x)$ satisfies Lemma 11.

To define the collection $\mathcal{C}_B$, we need some notations. Let $\mathcal{A}_B$ denote the subset of $\mathcal{B}_G(x)$ such that if $B' \in \mathcal{A}_B$, then B' is an ancestor of B in $\mathsf{B}_G^{\text{tree}}(x)$. Then we set $A_B = \cup_{B' \in \mathcal{A}_B} B'$ and denote C_B the connected component of $G - A_B$ containing B.

Algorithm 2: Proper chordal graph recognition

Input: A graph $G = (V, E)$;
Output: Decide if G is a proper chordal graph.

1 **foreach** $x \in V$ **do**
2   **if** *Algorithm 1 applied on G and x returns $\mathsf{B}_G^{\text{tree}}(x)$* **then**
3   **foreach** *block $B \in \mathcal{B}_G(x)$* **do**
4   let $C_1, \ldots, C_k$ be the connected components of $G[C_B] - B$;
5   **foreach** C_i *in* $C_1, \ldots, C_k$ **do**
6                  $\mathcal{N}_i \leftarrow \{X \subseteq B \mid \exists y \in C_i, X = N(y) \cap B\}$;
7   **end**
8   set $\mathcal{C}_B = \langle \mathcal{N}_1, \ldots, \mathcal{N}_k \rangle$ and $\mathcal{S}_B = \bigcup_{i \in [k]} \mathcal{N}_i$;
9   **end**
10   **if** $\forall B \in \mathcal{B}(x)$, $\mathcal{C}_B$ *is a collection of nested sets st.* $\mathsf{Nested\text{-}Convex}(\mathcal{C}_B, \mathcal{S}_B) \neq \emptyset$ **then**
11   **stop** and **return** *G is a proper chordal graph*;
12   **end**
13   **end**
14 **end**
15 **return** *G is not a proper chordal graph*;

Theorem 8. *We can decide in polynomial time whether a graph $G = (V, E)$ is proper chordal. Moreover, if G is proper chordal, an indifference tree-layout of G can be constructed in polynomial time.*

Proof. We prove that Algorithm 2 recognizes proper chordal graphs. Suppose first that G is proper chordal. Then for some vertex $x \in V$ there is an indifference tree-layout $\mathbf{T}_G = (T, r, \rho_G)$ rooted at $x = \rho^{-1}(r)$. By Lemma 12, Algorithm 1 computes $\mathcal{B}_G(x)$ and $\mathsf{B}_G^{\text{tree}}(x)$. Let $B \in \mathcal{B}_G(x)$. Then, we can additionally observe that for some vertex $y \in B$, $B = B_{\mathbf{T}_G}(y) \in \mathcal{B}_{\mathbf{T}_G}(G)$ and thereby $\mathcal{C}_B = \mathcal{C}_{\mathbf{T}_G}(y)$. Thus Lemma 11 implies that $\mathsf{Nested\text{-}Convex}(\mathcal{C}_B, \mathcal{S}_B) \neq \emptyset$. It follows that Algorithm 2 correctly concludes that G is a proper chordal graph.

Now suppose that Algorithm 2 concludes that G is a proper chordal graph. This implies that for some vertex $x \in V$: Algorithm 1 returns $\mathsf{B}_G^{\text{tree}}(x)$; and for every block $B \in \mathcal{B}(x)$, $\mathcal{C}_B$ is a collection of nested sets such that $\mathsf{Nested\text{-}Convex}(\mathcal{C}_B, \mathcal{S}_B) \neq \emptyset$. Now, using the proof of

Theorem 7 and the construction therein, it is possible to compute (in polynomial time) from $\mathsf{B}^{\mathsf{tree}}_G(x)$ and G, an indifference tree-layout of G rooted at x, certifying that G is proper chordal.

Finally observe that every step of Algorithm 2 (including Algorithm 1, see Lemma 12) requires polynomial time. $\qquad\square$

5.2 Isomorphism

We observe that, because an FPQ-hierarchy does not carry enough information to reconstruct the original graph, two non-isomorphic proper chordal graphs G and G' may share an FPQ-hierarchy (Figure 13) satisfying the conditions of Theorem 7. More precisely, given an FPQ-hierarchy of a proper chordal graph G that satisfies the conditions of Theorem 7, one can reconstruct an indifference tree-layout $\mathbf{T}$ of G. But $\mathbf{T}$ is not sufficient to test the adjacency between a pair of vertices. Indeed, for a given vertex y, we cannot retrieve $N(y) \cap A_{\mathbf{T}}(y)$ from $\mathsf{H}_G(x)$ since only the intersection of $N(y)$ with the parent block is present in $\mathsf{H}_G(x)$. In the example of Figure 13, the vertices w and w' are also adjacent to c and b, which do not belong to their parent block.

Let $\mathsf{H}_G(x)$ be the FPQ-hierarchy satisfying the conditions of Theorem 7, we define the *indifference FPQ-hierarchy*, denoted $\mathsf{H}^*_G(x)$, obtained from $\mathsf{H}_G(x)$ by adding to every skeleton edge e, a label $\hat{A}(e)$. Suppose that e is incident to the root of the FPQ-tree of the block B, then we set $\hat{A}(e) = |N(B) \cap A_{\mathbf{T}}(B)|$. We say that two indifference FPQ-hierarchies H^*_1 and H^*_2 are equivalent, denoted $\mathsf{H}^*_1 \approx^*_{\mathsf{FPQ}} \mathsf{H}^*_2$, if $\mathsf{H}_1 \approx_{\mathsf{FPQ}} \mathsf{H}_2$ and for every pair of mapped skeleton edges e_1 and e_2 we have $\hat{A}(e_1) = \hat{A}(e_2)$.

Let $\mathcal{S}_1 \in 2^{X_1}$ be a set of subsets of X_1 and $\mathcal{S}_2 \in 2^{X_2}$ be a set of subsets of X_2. We say that $\mathcal{S}_1$ and $\mathcal{S}_2$ are *isomorphic* if there exists a bijection $f : X_1 \to X_2$ such that $S_1 \in \mathcal{S}_1$ if and only if $S_2 = \{f(x) \mid x \in S_1\} \in \mathcal{S}_2$. For $S_1 \subseteq X_1$, we denote by $f(S_1) = \{f(y) \mid y \in S_1\}$.

Lemma 13. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two (connected) proper chordal graphs. Let $\mathsf{H}^*_1(x_1)$ be an indifference FPQ-hierarchy of G_1 and $\mathsf{H}^*_2(x_2)$ be an indifference FPQ-hierarchy of G_2. Then $\mathsf{H}^*_1(x_1) \approx^*_{\mathsf{FPQ}} \mathsf{H}^*_2(x_2)$ if and only if G_1 and G_2 are isomorphic with x_1 mapped to x_2.*

Proof. Suppose that f is a graph isomorphism between G_1 and G_2 such that $f(x_1) = x_2$. Let $\mathbf{T}_1 = (T, r, \rho)$ be an indifference tree-layout of G_1 such that $\rho^{-1}(r) = x_1$. Then as for every pair of vertices y and z, $yz \in E_1$ if and only if $f(y)f(z) \in E_2$, $\mathbf{T}_2 = (T, r, \rho \circ f)$ is an indifference tree-layout of G_2 such that $(\rho \circ f)^{-1}(r) = x_2$. We let $\mathsf{H}_1(x_1)$ and $\mathsf{H}_2(x_2)$ respectively denote the FPQ-hierarchy of G_1 and G_2 such that $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H}_1(x_1))$ contains $\mathbf{T}_1$ and $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H}_2(x_2))$ contains $\mathbf{T}_2$.

We also observe that $\mathcal{B}_{\mathbf{T}_1}(G_1)$ and $\mathcal{B}_{\mathbf{T}_2}(G_2)$ are isomorphic partitions of V_1 and V_2 respectively. It follows that the skeleton trees $\mathsf{B}^{\mathsf{tree}}_{G_1}(x_1)$ and $\mathsf{B}^{\mathsf{tree}}_{G_2}(x_2)$ are isomorphic trees. So if we consider $y_1 \in V_1$ such that $B_{\mathbf{T}_1}(y_1) \in \mathcal{B}_{\mathbf{T}_1}(G_1)$, then $B_{\mathbf{T}_2}(y_2) \in \mathcal{B}_{\mathbf{T}_2}(G_2)$ where $y_2 = f(y_1)$. Let $C_{\mathbf{T}_1}(x_1)$ be the connected component of $G_1 - A_{\mathbf{T}_1}(x_1)$ containing x_1. Then the connected component of $G_2 - A_{\mathbf{T}_2}(x_2)$ is the set $C_{\mathbf{T}_2}(x_2) = f(C_{\mathbf{T}_1}(x_1))$ and the collections $\mathcal{C}_{\mathbf{T}_1}(x_1)$ and $\mathcal{C}_{\mathbf{T}_2}(x_2)$ of nested sets are isomorphic. We set $\mathcal{S}_1 = \bigcup_{\mathcal{N}_1 \in \mathcal{C}_{\mathbf{T}_1}(x_1)} \mathcal{N}_1$ and $\mathcal{S}_2 = \bigcup_{\mathcal{N}_2 \in \mathcal{C}_{\mathbf{T}_2}(x_2)} \mathcal{N}$ and let denote T_1 an FPQ-tree such that $\mathfrak{S}_{\mathsf{T}_1} = \mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_1}(x_1), \mathcal{S}_1)$ and T_2 an FPQ-tree such that $\mathfrak{S}_{\mathsf{T}_2} = \mathsf{Nested\text{-}Convex}(\mathcal{C}_{\mathbf{T}_2}(x_2), \mathcal{S}_2)$. Then, as $\mathcal{S}_1$ and $\mathcal{S}_2$ are isomorphic, we have that $\mathsf{T}_1 \equiv_{\mathsf{FPQ}} \mathsf{T}_2$. In turn, this implies that $\mathsf{H}_1(x_1) \approx_{\mathsf{FPQ}} \mathsf{H}_2(x_2)$.
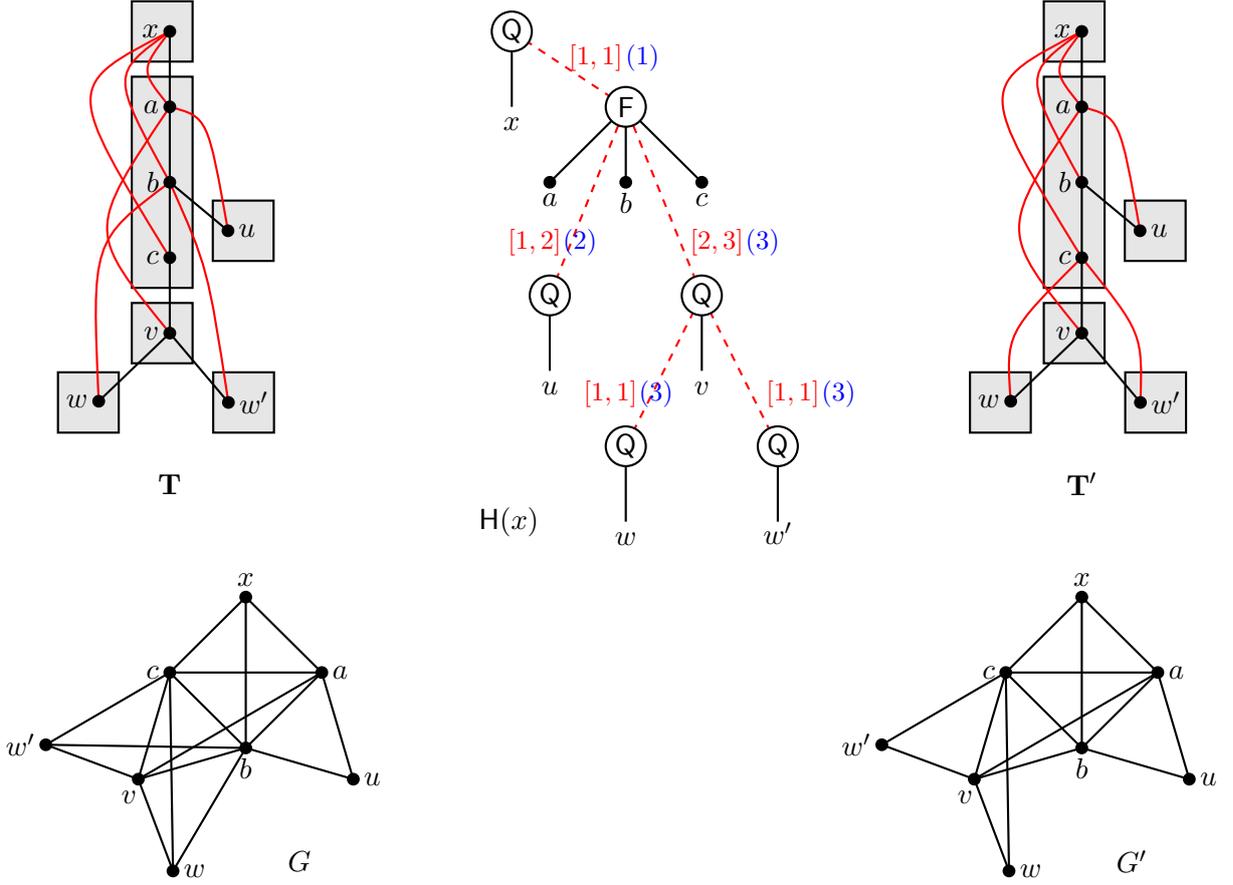
Figure 13: Two proper chordal graphs G and G' with their respective indifference tree-layouts $\mathbf{T}$ and $\mathbf{T}'$. We observe that G and G' are not isomorphic, but their respective skeleton trees $T_G(x)$ and $T_{G'}(x)$ are. Moreover, $\mathsf{H}(x)$ is an FPQ-hierarchy such that $\mathfrak{T}_{\mathsf{FPQ}}(\mathsf{H}(x))$ contains all the indifference tree-layouts rooted at x of G and of G'. We obtain $\mathsf{H}^*(x)$ for G by adding to $\mathsf{H}(x)$ the blue labels on the skeleton edges.

To conclude, we observe that as $\mathcal{B}_{\mathbf{T}_1}(G_1)$ and $\mathcal{B}_{\mathbf{T}_2}(G_2)$ are isomorphic partitions and $\mathsf{B}^{\mathsf{tree}}_{G_1}(x_1)$ and $\mathsf{B}^{\mathsf{tree}}_{G_2}(x_2)$ are isomorphic trees, for every skeleton edge e_1 of $\mathsf{B}^{\mathsf{tree}}_{G_1}(x_1)$, the label $A(e_1)$ is equal to the label $A(e_2)$ where e_2 is the corresponding skeleton edge of $\mathsf{B}^{\mathsf{tree}}_{G_2}(x_2)$. Thereby we can conclude that if G_1 and G_2 are isomorphic graphs, then $\mathsf{H}^*_1(x_1) \approx^*_{\mathsf{FPQ}} \mathsf{H}^*_2(x_2)$.

Suppose now that $\mathsf{H}^*_1(x_1) \approx^*_{\mathsf{FPQ}} \mathsf{H}^*_2(x_2)$. It follows that $\mathsf{H}_2(x_2)$ can be turned into an FPQ-hierarchy $\widetilde{\mathsf{H}}_2(x_2)$ isomorphic to $\mathsf{H}_1(x_1)$. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be respectively the trees that can be computed from $\widetilde{\mathsf{H}}_2(x_2)$ and $\mathsf{H}_1(x_1)$. By Claim 4 in the proof of Theorem 7, $\mathbf{T}_1$ and $\mathbf{T}_2$ are respectively indifference tree-layouts of G_1 and G_2. We observe that $\mathbf{T}_1$ and $\mathbf{T}_2$ are isomorphic rooted trees, respectively rooted at x_1 and x_2. Let f be the isomorphism from $\mathbf{T}_1$ to $\mathbf{T}_2$ We claim that f is an isomorphism from G_1 to G_2. Let y_1 and z_1 be two vertices of G_1. We denote $y_2 = f(y_1)$ and $z_2 = f(z_1)$ the corresponding vertices of G_2. First as $\mathbf{T}_1$ and $\mathbf{T}_2$ are tree-layouts, if $y_1 \not\prec_{\mathbf{T}_1} z_1$ and $z_1 \not\prec_{\mathbf{T}_1} y_1$, then $y_1 z_1 \notin E_1$. This implies that $y_2 z_2 \notin E_2$. Suppose without loss of generality that $y_1 \prec_{\mathbf{T}_1} z_1$, and hence $y_2 \prec_{\mathbf{T}_2} z_2$. As $\mathsf{H}^*_1(x_1) \approx^*_{\mathsf{FPQ}} \mathsf{H}^*_2(x_2)$, we have that $\mathcal{B}_{\mathbf{T}_1}(G_1)$ and $\mathcal{B}_{\mathbf{T}_2}(G_2)$ are isomorphic. It follows that y_1 and z_1 belong to the same block of

$\mathcal{B}_{\mathbf{T}_1}(G_1)$ if and only if y_2 and z_2 belong to the same block of $\mathcal{B}_{\mathbf{T}_2}(G_2)$. As blocks induce cliques, if y_1 and z_1 belong to the same block of $B_1 \in \mathcal{B}_{\mathbf{T}_1}(G_1)$, then $y_1z_1 \in E_2$ and $y_2z_2 \in E_2$. So suppose that the block of $B_1 \in \mathcal{B}_{\mathbf{T}_1}(G_1)$ containing z_1 does not contain y_1. Let $B_2 \in \mathcal{B}_{\mathbf{T}_2}(G_2)$ be the corresponding block that contains z_2 but not y_2. Let e_1, respectively e_2, be the skeleton edge of H_1, respectively H_2 incident to the root of the FPQ-tree of B_1, respectively of B_2. As $\mathsf{H}_1^*(x_1) \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^*(x_2)$, we have $\hat{A}(e_1) = \hat{A}(e_2)$. We observe that $y_1z_1 \in E_1$ if and only if the distance along the path from x_1 to z_1 in $\mathbf{T}_1$, between y_1 and z_1 is at most $\hat{A}(e_1) + |A_{\mathbf{T}_1}(z_1) \cap B_1|$. Likewise, $y_2z_2 \in E_2$ if and only if the distance along the path from x_2 to z_2 in $\mathbf{T}_2$, between y_2 and z_2 is at most $\hat{A}(e_2) + |A_{\mathbf{T}_2}(z_2) \cap B_2|$. Since $\mathbf{T}_1$ and $\mathbf{T}_2$ are isomorphic, and since $\mathcal{B}_{\mathbf{T}_1}(G_1)$ and $\mathcal{B}_{\mathbf{T}_2}(G_2)$ are isomorphic, this implies that $|A_{\mathbf{T}_1}(z_1) \cap B_1| = |A_{\mathbf{T}_2}(z_2) \cap B_2|$. It follows that $y_1z_1 \in E_1$ if and only if $y_2z_2 \in E_2$. We conclude that G_1 and G_2 are isomorphic graphs. $\qquad\square$

From Lemma 13, testing graph isomorphism on proper chordal graphs reduces to testing the equivalence between two indifference FPQ-hierarchies. To that aim, we use a similar approach to the one developed for testing interval graph isomorphism [37]. That is, we adapt the standard unordered tree isomorphism algorithm that assigns to every unordered tree a canonical *isomorphism code* [48, 1]. Testing isomorphism then amounts to testing equality between two isomorphism codes.

Let H^* be an indifference FPQ-hierarchy of a proper chordal graph $G = (V, E)$. Intuitively, the isomorphism code of H^* is a string obtained by concatenating information about the root node of H^* and the isomorphism codes of the sub-hierarchies rooted at its children. To guarantee the canonicity of the isomorphism code of H^*, some of the codes of these sub-hierarchies need to be sorted lexicographically. To that aim, we use the following convention:

$$\mathsf{L} <_{\mathsf{lex}} \mathsf{F} <_{\mathsf{lex}} \mathsf{P} <_{\mathsf{lex}} \mathsf{Q} <_{\mathsf{lex}} 0 <_{\mathsf{lex}} 1 \ldots <_{\mathsf{lex}} n <_{\mathsf{lex}} \ldots,$$

Moreover the separating symbols (such as brackets, commas. . .) used in the isomorphism code for the sake of readability are irrelevant for the sort.

Before formally describing the isomorphism code of H^*, let us remind that, in an indifference FPQ-hierarchy, we can classify the children of any node t in two categories: we call a node t' a *skeleton child* of t if the tree edge $e = tt'$ is a skeleton edge of H^*, otherwise we call it a *block child* of t. We observe that the block children of a node t belong with t to the FPQ-tree of some block of $\mathcal{B}_G(x)$. It follows from the definition of an FPQ-tree, that the block children of a given node t are ordered and depending on the type of t, these nodes can be reordered. On the contrary, the skeleton children of a node t are not ordered.

For every node t of H^*, we define a code, denoted $\mathsf{code}(t)$. We will define the isomorphism code of H^* as $\mathsf{code}(\mathsf{H}^*) = \mathsf{code}(r)$, where r is the root node of H^*. We let $b_1, \ldots, b_k$ denote the block children of node t (if any, and ordered from 1 to k) and $s_1, \ldots, s_\ell$ denote the skeleton children of t (if any). For a node t, the set of *eligible permutations* of the indices $[1, k]$ of its chidlren depends on $\mathsf{type}(t)$:

- if $\mathsf{type}(t) = \mathsf{F}$, then the identity permutation is the unique eligible permutation;

- if $\mathsf{type}(t) = \mathsf{P}$, then every permutation is eligible;

- if $\mathsf{type}(t) = \mathsf{Q}$, then the identity or its reverse permutation are the two eligible permutations.

The code of t, denoted $\mathsf{code}(t)$, is obtained by minimizing with respect to $<_{\mathsf{lex}}$ over all eligible permutations β of t:

$$
\mathsf{code}(t, \beta) = \left\{
\begin{array}{l}
\mathsf{size}(t) \circ \mathsf{type}(t) \circ \\
\quad\quad \mathsf{code}(b_{\beta(1)}) \circ \cdots \circ \mathsf{code}(b_{\beta(k)}) \circ \\
\quad\quad\quad\quad \mathsf{label}(s_{\pi_\beta(1)}, \beta) \circ \mathsf{code}(s_{\pi_\beta(1)}) \circ \cdots \circ \mathsf{label}(s_{\pi_\beta(\ell)}, \beta) \circ \mathsf{code}(s_{\pi_\beta(\ell)})
\end{array}
\right.
$$

where:

- $\mathsf{type}(t) \in \{\mathsf{L}, \mathsf{F}, \mathsf{P}, \mathsf{Q}\}$, indicates whether t a leaf (L), a F-node, a P-node, or a Q-node.

- $\mathsf{size}(t) \in \mathbb{N}$, stores the number of nodes in the sub-hierarchy rooted at t (including t).

- $\mathsf{label}(s, \beta)$, with s being a skeleton child s of t and β being a permutation of $[1, k]$. Let e be the skeleton edge of H^* between s and t. If $I(e) = [a, b]$, we set $I^c(e) = [k+1-b, k+1-a]$. Then, we set $\mathsf{label}(s) = \langle I^\beta(e), A(e) \rangle$, where

$$
I^\beta(e) = \left\{
\begin{array}{ll}
I(e), & \beta \text{ is the identity permutation} \\
I^c(e), & \text{otherwise.}
\end{array}
\right.
$$

- π_β is, for some permutation β of $[1, k]$, a permutation of $[1, \ell]$ that minimizes, with respect to $<_{\mathsf{lex}}$:
$$
\mathsf{label}(s_{\pi_\beta(1)}, \beta) \circ \mathsf{code}(s_{\pi_\beta(1)}) \circ \cdots \circ \mathsf{label}(s_{\pi_\beta(\ell)}, \beta) \circ \mathsf{code}(s_{\pi_\beta(\ell)}).
$$

Remark. Observe that for a P-node t, the permutation π_β is independant of the choice of β. Indeed, thanks to Lemma 3, $I(e) = [1, k]$ and thereby $I^\beta(e) = I(e)$. So, if t is an F-node or a P-node, then $\mathsf{code}(t)$ can be greedily computed in polynomial time. If t is a Q-node, $\mathsf{code}(t)$ can still be computed in polynomial time by comparing $\mathsf{code}(t, \beta)$ and $\mathsf{code}(t, \overline{\beta})$ where β is the identity permutation and $\overline{\beta}$ its reverse.

Lemma 14. *Let H_1^* and H_2^* be indifference FPQ-hierarchies of the graphs G_1 and G_2 respectively. Then $\mathsf{H}_1^* \approx_{FPQ}^* \mathsf{H}_2^*$ if and only if $\mathsf{code}(\mathsf{H}_1^*) = \mathsf{code}(\mathsf{H}_2^*)$.*

Proof. Let H^* be an indifference FPQ-hierarchy. We claim that given $\mathsf{code}(\mathsf{H}^*)$, one can deterministically construct an FPQ-hierarchy $\mathsf{H}^{*(rep)}$ such that $\mathsf{H}^* \approx_{FPQ}^* \mathsf{H}^{*(rep)}$. To that aim, we need to parse $\mathsf{code}(\mathsf{H}^*)$ in order to identify the children of each node. This can be done using the $\mathsf{size}(\cdot)$ information. The smallest possible indifference FPQ-hierarchy H^* contains a Q-node as root and one leaf. It can clearly be reconstructed from $\mathsf{code}(\mathsf{H}^*) = 2\mathsf{Q}1\mathsf{L}$. So suppose that the property is true for every indifference FPQ-hierarchy containing i nodes with $2 \leq i \leq n$ and assume that H^* contains $n+1$ nodes. Let r be the root of H^*. Then $\mathsf{size}(r) \circ \mathsf{type}(r) \circ \mathsf{size}(c_1)$, where c is the first child of r is a prefix of $\mathsf{code}(\mathsf{H}^*)$. Then the second child c_2 of r can be identified after reading $\mathsf{size}(c_1)$ times a $\mathsf{type}(\cdot)$ tag in $\mathsf{code}(\mathsf{H}^*)$. So every child of r can be identified. Moreover, it is possible to distinguish whether a child c_j of r is a block child or a skeleton child by the presence or not of the string $\mathsf{label}(c_j)$ delimited with the special characters $\langle \cdot \rangle$ and to assign each of them the correct $\mathsf{label}(\cdot)$ information. Finally, observe that by construction of $\mathsf{code}(\mathsf{H}^*)$ the block children of r are permuted in a way compatible with its type (F, P or Q). As every child of r is the root of a sub-hierarchy containing at most n nodes, by the recursive hypothesis, we can conclude that the returned FPQ-hierarchy $\mathsf{H}^{*(rep)}$ verifies $\mathsf{H}^* \approx_{FPQ}^* \mathsf{H}^{*(rep)}$.

From the above discussion, if $\mathsf{code}(\mathsf{H}_1^*) = \mathsf{code}(\mathsf{H}_2^*)$, then $\mathsf{H}_1^{*(rep)}$ and $\mathsf{H}_2^{*(rep)}$ are isomorphic FPQ-hierarchies (in terms of labeled ordered trees). As $\mathsf{H}_1^* \approx_{\mathsf{FPQ}}^* \mathsf{H}_1^{*(rep)}$ and $\mathsf{H}_2^* \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^{*(rep)}$, we obtain that $\mathsf{H}_1^* \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^*$.

Suppose now that $\mathsf{H}_1^* \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^*$. Consider $\mathsf{H}_1^{*(rep)}$ and $\mathsf{H}_2^{*(rep)}$ respectively computed from $\mathsf{code}(\mathsf{H}_1^*)$ and $\mathsf{code}(\mathsf{H}_2^*)$. It follows that $\mathsf{H}_1^{*(rep)} \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^{*(rep)}$. Now observe that by construction $\mathsf{code}(\mathsf{H}_1^*) = \mathsf{code}(\mathsf{H}_1^{*(rep)})$ and henceforth among the set of FPQ-hierarchies equivalent to H_1^*, $\mathsf{code}(\mathsf{H}_1^{*(rep)})$ is the smallest lexicographic one. The same holds for H_1^* and $\mathsf{H}_1^{*(rep)}$. Consequently, if $\mathsf{code}(\mathsf{H}_1^*) \neq \mathsf{code}(\mathsf{H}_2^*)$, then one is smaller lexicographically than the other: a contradiction. $\quad\square$

Theorem 9. *Let G_1 and G_2 be two proper chordal graphs. One can test in polynomial time if $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic graphs.*

Proof. The algorithm is working as follows. First, compute a tree-layout $\mathbf{T}_1$ of G_1 and the indifference FPQ-hierarchy H_1^* such that $\mathbf{T}_1 \in \mathfrak{T}_{\mathsf{FPQ}}(H_1)$. This can be done in polynomial time by Theorem 8. Then for every vertex $x_2 \in V_2$, we test if there exists an indifference tree-layout $\mathbf{T}_2$ rooted at x_2; compute the corresponding indifference FPQ-hierarchy H_2^* and test whether $\mathsf{H}_1^* \approx_{\mathsf{FPQ}}^* \mathsf{H}_2^*$. By Lemma 14, testing equivalence between FPQ-hierarchies can be done by computing and comparing the isomorphism codes of H_1^* and H_2^*. Moreover, this latter task can be achieved in polynomial time. By Lemma 13, if one of these tests is positive, then we can conclude that G_1 and G_2 are isomorphic graphs. $\quad\square$

6 Conclusion

Our results demonstrate that proper chordal graphs form a rich class of graphs. First, its relative position with respect to important graph subclasses of chordal graphs and the fact that the isomorphism problem belongs to P for proper chordal graphs shows that they form a non-trivial potential island of tractability for many other algorithmic problems. In this line, we let open the status of Hamiltonian cycle, which is polynomial time solvable in proper interval graphs [4, 31] and interval graphs [32, 5], but NP-complete on strongly chordal graphs [39]. We were only able to resolve the special case of split proper chordal graphs. An intriguing algorithmic question is whether proper chordal graphs can be recognized in linear time. Second, the canonical representation we obtained of the set of indifference tree-layouts rooted at some vertex witnesses the rich combinatorial structure of proper chordal graphs. We believe that this structure has to be further explored and could be important for the efficient resolution of more computational problems. For example, as proper chordal graphs form a hereditary class of graphs, one could wonder if the standard graph modification problems (vertex deletion, edge completion or deletion, etc.), which are NP-complete by [34], can be resolved in FPT time. The structure of proper chordal graphs is not yet fully understood. The first natural question on this aspect is to provide a forbidden induced subgraph characterization. This will involve infinite families of forbidden subgraphs. Furthermore, understanding what makes a vertex the root of an indifference tree-layout is certainly a key ingredient for a fast recognition algorithm. We would like to stress that a promising line of research is to consider further tree-layout based graph classes. For this, following the work of Damaschke [12], Hell et al. [29] and Feuilloley and Habib [17] on layouts, we need to investigate in a more systematic way various patterns to exclude, including rooted tree patterns.

References

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

[2] S. Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Science*, 45(11):1607–1620, 1957. `doi:10.1073/pnas.45.11.1607`.

[3] C. Berge. Färbung von graphen deren sämtliche beziehungsweise deren ungerade kreise starr sind (zusammenfassung). *Wissenschaftliche Zeitschrift, Martin Luther Univ. Halle-Wittenberg, Math.-Naturwiss, Reihe*, pages 114–115, 1961.

[4] A.A. Bertossi. Finding Hamiltonian circuits in proper interval graphs. *Information Processing Letters*, 17:97–101, 1983. `doi:10.1016/0020-0190(83)90078-9`.

[5] A.A. Bertossi and M.A. Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Information Processing Letters*, 23:195–200, 1986. `doi:10.1016/0020-0190(86)90135-3`.

[6] D. Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory, Series B*, 49(1):103-136, 1990. `doi:10.1016/0095-8956(90)90066-9`.

[7] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithm. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[8] S. Chaplick. Intersection Graphs of Non-crossing Paths. *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*. Volume 11789 of Lecture Notes in Computer Science, pages 311–324, 2019. `doi:10.1007/978-3-030-30786-8`.

[9] M. Chein, M. Habib, and M.-C. Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37:35–50, 1981. `doi:10.1016/0012-365X(81)90138-2`.

[10] D.G. Corneil, H. Lerchs, and L.K. Stewart-Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(1):163–174, 1981. `doi:10.1016/0166-218X(81)90013-5`.

[11] W.H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3):734–765, 1980. `doi:10.4153/CJM-1980-057-7`.

[12] Peter Damaschke. Forbbiden ordered subgraphs. *Topics in Combinatorics and Graph Theory*, pages 219–229, 1990. `doi:10.1007/978-3-642-46908-4_25`.

[13] G. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961.

[14] P. Duchet. Classical perfect graphs: An introduction with emphasis on triangulated and interval graphs. In C. Berge and V. Chvátal, editors, *Topics on Perfect Graphs*, volume 88 of *North-Holland Mathematics Studies*, pages 67–96. North-Holland, 1984. `doi:10.1016/S0304-0208(08)72924-4`.

[15] Dwight Duffus, Mark Ginn, and Vojtěch Rödl. On the computational complexity of ordered subgraph recognition. *Random Structure and Algorithms*, 7(3):223–268, 1995. `doi:10.1002/rsa.3240070304`.

[16] M. Farber. Characterizations of strongly chordal graphs. *Discrete Mathematics*, 43(2-3):173–189, 1983. `doi:10.1016/0012-365X(83)90154-1`.

[17] Laurent Feuilloley and Michel Habib. Graph classes and forbidden patterns on three vertices. *SIAM Journal on Discrete Mathematics*, 35(1):55–90, 2021. `doi:10.1137/19M1280399`.

[18] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1):25–66, 1967. `doi:10.1007/BF02020961`.

[19] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory Series B*, 16:47–56, 1974. `doi:10.1016/0095-8956(74)90094-X`.

[20] A. Ghouilà-Houri. Caractérisation des graphes non-orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. *Compte-Rendus de l'Académie des Sciences*, 254:1370–1371, 1962.

[21] M. Ginn. Forbidden ordered subgraph vs. forbidden subgraph characterizations of graph classes. *Journal of Graph Theory*, 30(71-76), 1999. `doi:10.1002/(SICI)1097-0118(199902)30:2<71::AID-JGT1>3.0.CO;2-G`.

[22] M.C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978. `doi:10.1016/0012-365X(78)90178-4`.

[23] M.C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980. `doi:10.1016/C2013-0-10739-8`.

[24] S. Guzmán-Pro, P. Hell, and C. Hernández-Cruz. Describing hereditary properties by forbidden circular orderings. *Applied Mathematics and Computation*, 438:127555, 2023. `doi:10.1016/j.amc.2022.127555`.

[25] M. Habib, R.M. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000. `doi:10.1016/S0304-3975(97)00241-7`.

[26] M. Habib and C. Paul. A survey on algorithmic aspects of modular decomposition. *Computer Science Review*, 4:41–59, 2010. `doi:10.1016/j.cosrev.2010.01.001`.

[27] A. Hajnal and J. Surányi. Über die Auflösung von Graphen in vollständige Teilgraphen. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Sectio Mathematica*, 1:113–121, 1958.

[28] G. Hajös. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11, 1957. Problem 65.

[29] Pavol Hell, Bojan Mohar, and Arash Rafiey. Ordering without forbidden patterns. In *Annual European Symposium on Algorithms (ESA)*, volume 8737 of *Lecture Notes in Computer Science*, pages 554–565, 2014. `doi:10.1007/978-3-662-44777-2\_46`.

[30] W.-L. Hsu. $o(n.m)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995. `doi:10.1137/s0097539793260726`.

[31] L. Ibarra. A simple algorithm to find hamiltonian cycles in proper interval graphs. *Information Processing Letters*, 109:1105–1108, 2009. `doi:10.1016/j.ipl.2009.07.010`.

[32] J.M. Keil. Finding Hamiltonian circuits in interval graphs. *Information Processing Letters*, 20:201–206, 1985. `doi:10.1016/0020-0190(85)90050-X`.

[33] H. Lerchs. On cliques and kernels. Technical report, Departement of Computer Science, University of Toronto, 1971.

[34] J. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. `doi:10.1016/0022-0000(80)90060-4`.

[35] G. Liotta, I. Rutter, and A. Tappini. Simultaneous FPQ-ordering and hybrid planarity testing. *Theoretical Computer Science*, 874:59–79, 2021. `doi:10.1016/j.tcs.2021.05.012`.

[36] P. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Computers and Mathematics with Applications*, 25(7):15–25, 1993. `doi:10.1016/0898-1221(93)90308-I`.

[37] G.S. Lueker and K.S. Booth. A linear time algorithm for deciding interval graphs isomorphism. *Journal of ACM*, 26(2):183–195, 1979. `doi:10.1145/322123.322125`.

[38] C.L. Monma and V.K .Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41:141–181, 1986. `doi:10.1016/0095-8956(86)90042-0`.

[39] H. Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156:291–298, 1996. `doi:10.1016/0012-365X(95)00057-4`.

[40] J. Nešetřil and P. Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27:6:1022-1041, 2006. `doi:10.1016/j.ejc.2005.01.010`.

[41] S. Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37:21–25, 1991. `doi:10.1016/0020-0190(91)90245-D`.

[42] F.S. Roberts. *Representations of indifference relations*. PhD thesis, Standford University, 1968.

[43] F.S. Roberts. Indifference graphs. In F. Harrary, editor, *Proof Techniques in Graph Theory*, pages 139–146, 1969.

[44] D. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(597-609), 1970. `doi:10.1016/0022-247X(70)90282-9`.

[45] Dale J. Skrien. A relationship between triangulated graphs, comparability graphs, proper interval graphs, proper circular-arc graphs, and nested interval graphs. *Journal of Graph Theory*, 6:309–316, 1982. `doi:10.1002/jgt.3190060307`.

[46] D.P. Sumner. Graphs indecomposable with respect to the X-join. *Discrete Mathematics*, 6:281–298, 1973. `doi:10.1016/0012-365X(73)90100-3`.

[47] R. Uehara, S. Toda, and T. Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145:479–482, 2005. `doi:doi:10.1016/j.dam.2004.06.008`.

[48] G. Valiente. *Algorithms on trees and graphs.* Texts in Computer Science. Springer, 2002. `doi:10.1007/978-3-030-81885-2`.