



# Tree-Layout Based Graph Classes: Proper Chordal Graphs

Christophe Paul   

CNRS, Université Montpellier, France

Evangelos Protopapas  

CNRS, Université Montpellier, France

---

## Abstract

Many important graph classes are characterized by means of layouts (a vertex ordering) excluding some patterns. For example, a graph  $G = (V, E)$  is a proper interval graph if and only if  $G$  has a layout  $\mathbf{L}$  such that for every triple of vertices such that  $x \prec_{\mathbf{L}} y \prec_{\mathbf{L}} z$ , if  $xz \in E$  and  $yz \in E$ . Such a triple  $x, y, z$  is called an *indifference triple*. In this paper, we investigate the concept of excluding a set of patterns in *tree-layouts* rather than 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  $\rho_G$  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  $x$ . Excluding patterns in a tree-layout is now defined using the ancestor relation. This leads to an unexplored territory of graph classes. In this paper, we initiate the study of such graph classes with the class of *proper chordal graphs* defined by excluding indifference triples in tree-layouts. Our results combine characterization, compact and canonical representation as well as polynomial time algorithms for the recognition and the graph isomorphism of proper chordal graphs. For this, one of the key ingredients is the introduction of the concept of FPQ-hierarchy generalizing the celebrated PQ-tree data-structure.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Graph algorithms; Mathematics of computing  $\rightarrow$  Combinatorial algorithms; Mathematics of computing  $\rightarrow$  Graph theory; Theory of computation  $\rightarrow$  Data structures design and analysis; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Graph classes, Graph representation, Graph isomorphism

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2024.55

**Related Version** *Full Version*: <https://arxiv.org/abs/2211.07550>

**Funding** *Christophe Paul*: Research supported by ANR-DFG project UTMA ANR-20-CE92-0027.

*Evangelos Protopapas*: Research supported by ANR-DFG project UTMA ANR-20-CE92-0027.

## 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. If  $\mathcal{F} = \{H\}$ , then we simply say that  $G$  is  *$H$ -free* if  $H \not\subseteq_i G$ . Important graph classes are characterized by a finite set  $\mathcal{F}$  of minimal forbidden subgraphs. A popular example is the class of cographs [32, 45]. 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 [32, 11]. As witnessed by *chordal graphs* [26, 3], not every hereditary graph family is characterized by excluding a finite set of



© Christophe Paul and Evangelos Protopapas;  
licensed under Creative Commons License CC-BY 4.0

41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024).

Editors: Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshantov;

Article No. 55; pp. 55:1–55:18



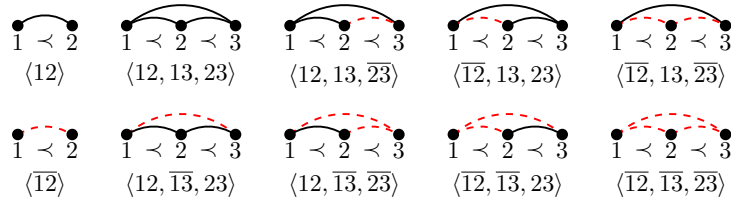
Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



minimal forbidden subgraphs: a graph is chordal if it does not contain a chordless cycle of length at least 4 as an induced subgraph.

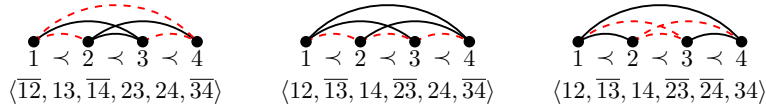
To circumvent this issue, Skrien [44] and Damaschke [13] proposed to embed graphs in some additional structure such as vertex orderings, also called *layouts*. An *ordered graph* is then defined as a pair  $(G, \prec_G)$  such that  $\prec_G$  is a total ordering of the vertex set  $V$  of the graph  $G = (V, E)$ . 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  $(H, \prec_H) \not\subseteq_p (G, \prec_G)$ . 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  $(H, \prec_H) \in \mathcal{P}$ ,  $(H, \prec_H) \not\subseteq_p (G, \prec_G)$ . We let  $\mathcal{L}(\mathcal{P})$  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 10 patterns on at most 3 vertices.  $\mathcal{L}(\langle \overline{12}, 13, 23 \rangle)$  is the class of chordal graphs.

Interestingly, it is known that chordal graphs are characterized by excluding a unique pattern:  $\mathcal{P}_{\text{chordal}} = \{ \langle \overline{12}, 13, 23 \rangle \}$ , see Figure 1 [13, 15]. This characterization relies on the fact that a graph is chordal if and only if it admits a *simplicial elimination ordering*<sup>1</sup> [14, 43]. Ginn [20] proved 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: cographs are characterized as  $P_4$ -free graphs (see discussion above) but needs a set  $\mathcal{P}_{\text{cograph}}$  of several excluded patterns (see Figure 2) to be characterized [13]:

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



■ **Figure 2** The three size 4 forbidden patterns of cographs.

In [16], 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

<sup>1</sup> 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  $G[\{y \in V \mid y \prec_G x\}]$ .

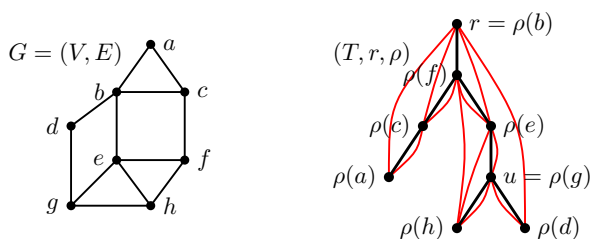
its complement is 2-connected. Hell et al. [28] have recently shown that if  $\mathcal{P}$  only contains patterns of size at most 3, then  $\mathcal{L}(\mathcal{P})$  can be recognized in polynomial time using a 2-SAT approach. Besides chordal graphs (see discussion above), these graph classes comprise very important graph classes, among others:

- *Interval graphs* [27, 2, 22] exclude  $\mathcal{P}_{\text{int}} = \{\langle \overline{12}, 13, \overline{23} \rangle, \langle \overline{12}, 13, 23 \rangle\}$  [40]: 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  $\mathcal{P}_{\text{int}}$ -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* [41, 42] exclude  $\mathcal{P}_{\text{proper}} = \mathcal{P}_{\text{int}} \cup \{\langle 12, 13, \overline{23} \rangle\}$ . 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). This characterization of proper interval graphs as  $\mathcal{L}(\mathcal{P}_{\text{proper}})$  follows from the existence of the so-called *indifference orderings* that are exactly the layouts excluding  $\mathcal{P}_{\text{proper}}$  [41, 42].
- *Trivially perfect graphs* [21] exclude  $\mathcal{P}_{\text{trivPer}} = \mathcal{P}_{\text{chordal}} \cup \{\langle 12, \overline{13}, 23 \rangle\}$ . 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  $P_4$  and  $C_4$  contains one of the patterns of  $\mathcal{P}_{\text{trivPer}}$  (see [17]).

For more examples, the reader should refer to [13, 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.

**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. [23] 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  $\rho_G : V \rightarrow V_T$  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  $(T, r, \rho)$  of a graph  $G = (V, E)$ .

Let  $\mathbf{T}_G = (T, r, \rho_G)$  be a *tree-layout* of a graph  $G$ . We observe that, from Definition 1,  $T$  is not a *Trémaux tree* since it is not necessarily a spanning tree of  $G$  (see [39] and [6] for similar concepts). It is easy to see that if  $T$  is a path, then  $\mathbf{T}_G$  defines 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

$\mathbf{T}_G = (T, r, \rho_G)$  of a graph  $G$  is defined similarly as excluding a pattern in a layout, but now using the ancestor relation  $\prec_T$ . 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  $\mathfrak{T}((H, \mathbf{L}_H))$ . 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.

► **Observation 2.** *The class  $\mathfrak{L}(\langle \overline{12} \rangle)$  is the set of complete graphs while the class  $\mathfrak{T}(\langle \overline{12} \rangle)$  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 the unexplored territory composed by graph classes defined by excluding a set of patterns in a tree-layout.

**Our contributions.** As a first case study, we consider the patterns characterizing interval graphs and proper interval graphs. We first show that if we consider the interval graphs patterns  $\mathcal{P}_{\text{int}}$ , 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  $\mathfrak{T}(\mathcal{P}_{\text{int}})$  (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). It is known that  $K_{1,3}$  is an interval graph but not a proper one. Following this line, Gavril [19], in his seminal paper characterizing chordal graphs as the intersection graphs of a subset of subtrees of a tree, considered the class of intersection graphs of a set of *proper* subtrees of a tree (no subtree is contained in 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. In a recent paper, Chaplick [9] investigated this question and considered the class of intersection graphs of non-crossing paths in a tree.

Our main contribution is to propose a natural definition of *proper chordal graphs* by means of forbidden patterns on tree-layouts: a graph is proper chordal if it belongs to  $\mathfrak{T}(\mathcal{P}_{\text{proper}})$ , the class of graphs admitting a  $\mathcal{P}_{\text{proper}}$ -free tree-layout, hereafter called *indifference tree-layout*. Recall that  $\mathcal{P}_{\text{proper}}$  are the patterns characterizing proper interval graphs on layouts. It can be observed that proper chordal graphs and intersection graphs of non-crossing paths in a tree are incomparable graph classes. Table 1 resumes the discussion above.

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

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

We then provide a thorough study of the combinatorial and algorithmic aspects of proper chordal graphs. 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 may have multiple (proper) interval models. For a given graph, these interval models are all captured in a canonical representation encoded by the celebrated PQ-tree data-structure [7]. We show that the set of indifference tree-layouts can also be represented

in a canonical and compact way by means of a tree data-structure generalizing PQ-trees, that we call FPQ-hierarchies, see Theorem 11. These structural results have very interesting algorithmic implications. First, we can design a polynomial time recognition algorithm for proper chordal graphs, see Theorem 14. Second, we show that the isomorphism problem restricted to proper chordal graphs is polynomial time solvable, see Theorem 16. Interestingly, this problem is GI-complete on (strongly) chordal graphs [38, 46]. So considering proper chordal graphs allows us to push the tractability further towards its limit. We believe that beyond proper chordal graphs, the concept of FPQ-hierarchy is interesting on its own as it is strongly related to the concept of (weakly) partitive families [12, 10] and thereby the theory of modular decomposition [37, 25].

## 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  $xy$  denote 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  $N(x) = \{y \in V \mid xy \in E\}$ . The closed neighbourhood of a vertex  $x$  is  $N[x] = N(x) \cup \{x\}$ . Let  $S$  be a subset of vertices of  $V$ . The graph resulting from the removal of  $S$  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<sup>2</sup> 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  $\text{lca}_{(T,r)}(u, v)$ . 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  $u$ .

**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  $\sigma_{(T,r)}^v$  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  $v = \text{lca}_{(T,r)}(x, y)$  respectively being an ancestor of  $x$  and of  $y$ . Then  $x \prec_{\sigma_{(T,r)}} y$  if and only if  $u_x \prec_{\sigma_{T,r}^v} u_y$ .

<sup>2</sup> To avoid confusion, we reserve the term *vertex* for graphs and *node* for trees.

**Tree-layouts of graphs.** Let  $\mathbf{T} = (T, r, \rho)$  be a tree-layout of a graph  $G = (V, E)$  (see Definition 1). 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 A novel characterization of chordal graphs

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  $\mathbf{M}_G^{\mathbf{T}} = (T, \mathcal{T}, \tau_G)$  where  $T$  is a tree,  $\mathcal{T}$  is a family of subtrees of  $T$  and  $\tau_G : V \rightarrow \mathcal{T}$  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  $T^x = \tau_G(x)$ . Likewise, an intersection model of an interval graph  $G$  is  $\mathbf{M}_G^{\mathbf{I}} = (P, \mathcal{P}, \tau_G)$  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  $P^x$ . So, chordal graphs clearly appear as the tree-like version of interval graphs. We prove that this similarity can also be observed when characterizing these graph classes by means of forbidden patterns. Recall that the class of interval graphs is  $\mathfrak{L}(\mathcal{P}_{\text{int}})$  where  $\mathcal{P}_{\text{int}} = \{\langle \overline{12}, 13, 23 \rangle, \langle \overline{12}, 13, \overline{23} \rangle\}$ .

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

## 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  $\mathbf{M}_G^{\mathbf{I}}$  such that for every pair of intervals none is a subinterval of another [41, 42]. In terms of a pattern characterization, we have seen that the class of proper interval graphs is  $\mathfrak{L}(\mathcal{P}_{\text{proper}})$  where  $\mathcal{P}_{\text{proper}} = \{\langle \overline{12}, 13, 23 \rangle, \langle \overline{12}, 13, \overline{23} \rangle, \langle 12, 13, \overline{23} \rangle\}$  [42, 13]. Hereafter a layout that is  $\mathcal{P}_{\text{proper}}$ -free is called an *indifference layout*. Indifference layouts have several characterizations, see Theorem 4 below.

► **Theorem 4.** [35, 41] *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  $N(x) \cap D_{\mathbf{L}_G}(y) \subseteq N(y) \cap D_{\mathbf{L}_G}(y)$ .

**Proper chordal graphs.** To understand what are the *tree-like* proper interval graphs, we propose the following definition.

► **Definition 5.** *A graph  $G = (V, E)$  is a proper chordal graph if  $G \in \mathfrak{L}(\mathcal{P}_{\text{proper}})$ .*

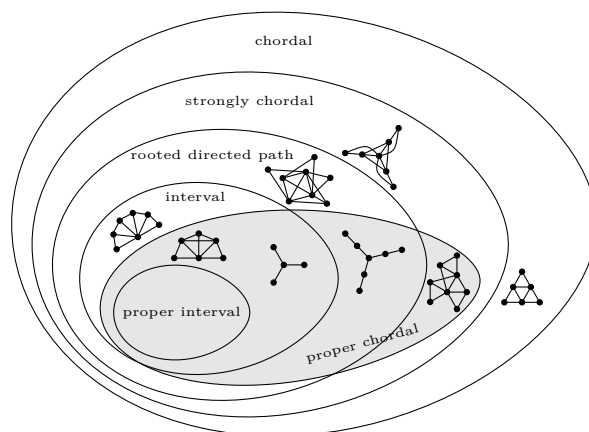
Hereafter, a tree-layout  $\mathbf{T}_G$  of a graph  $G$  that is  $\mathcal{P}_{\text{proper}}$ -free will be called an *indifference tree-layout*. We first prove that Theorem 4 generalizes to indifference tree-layouts.

► **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  $N(x) \cap D_{\mathbf{T}_G}(y) \subseteq N(y) \cap D_{\mathbf{T}_G}(y)$ .



Clearly, as an indifference layout is an indifference tree-layout, every proper interval graph is a proper chordal graph. Also, as  $\mathcal{P}_{\text{chordal}} \subset \mathcal{P}_{\text{proper}}$ , proper chordal graphs are chordal graphs. However this inclusion is strict as  $k$ -suns, for  $k \geq 3$ , are not proper chordal. More generally, Figure 4 positions proper chordal graphs with respect to important subclasses of chordal graphs, see [8] for definitions of these classes.



■ **Figure 4** Relationship between proper chordal graphs and subclasses of chordal graphs.

### 3 FPQ-trees and FPQ-hierarchies

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

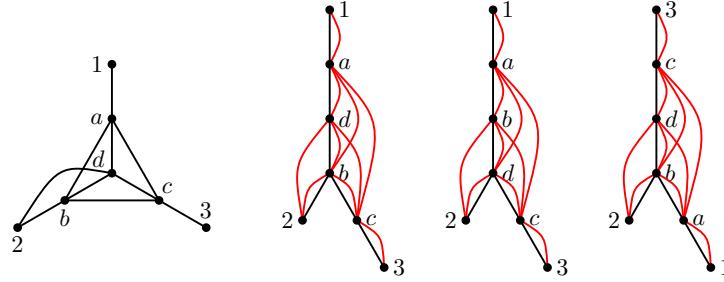
Two vertices  $x$  and  $y$  of a graph  $G$  are *true-twins* if  $N[x] = N[y]$ . It is easy to see that the true-twin relation is an equivalence relation. If  $G$  contains some true-twins, then, by Theorem 4, the vertices of any equivalence class occurs consecutively (and in arbitrary order) in an 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.

In the case of interval graphs, the set of intersection models, and hence of  $\mathcal{P}_{\text{int}}$ -free layouts, is structured by means of *modules* [18], in a similar way to the true-twin equivalence classes for  $\mathcal{P}_{\text{proper}}$ -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 [29] 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* [10] and can thereby be represented through a linear size tree, called the *modular decomposition tree* (see [25] 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 [10, 12]. 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.

As shown by Figure 5, a proper chordal graph can also have several indifference tree-layouts. In order to represent the set of  $\mathcal{P}_{\text{proper}}$ -tree-layouts of a given graph, we will define a structure called *FPQ-hierarchies*, based on FPQ-trees [34], a variant of PQ-trees.



■ **Figure 5** 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  $\mathcal{L}(T)$  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  $T \equiv_{\text{FPQ}} T'$ , if one can be turned into a labelled tree isomorphic to the other by a series of the following two operations: *permute*( $u$ ) which permutes in any possible way the children of a P-node  $u$ ; and *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  $\mathfrak{S}_{\text{FPQ}}(T)$  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* [10, 12]. As a consequence, we have the following property, which was also proved in [36].

► **Lemma 7.** [10, 12, 36] *Let  $\mathfrak{S}_T$  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  $\mathfrak{S}_T$  if and only if there exists an internal node  $u$  of  $T$  such that*

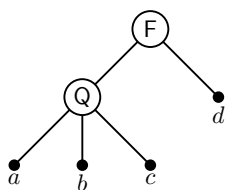
- *either  $I = \mathcal{L}_T(u)$ ;*
- *or  $u$  is a Q-node and there exists a set of children  $v_1, \dots, v_s$  of  $u$  that are consecutive in  $<_{T,u}$  and such that  $I = \bigcup_{1 \leq i \leq s} \mathcal{L}_T(v_i)$ .*

*We observe that  $u = \text{lca}_T(I)$ .*

Given a set  $\mathcal{S} \subseteq 2^X$  of subsets of the ground set  $X$ , we let  $\text{Convex}(\mathcal{S})$  denote the set of permutations of  $X$  such that for every  $S \in \mathcal{S}$ ,  $S$  is a factor of  $\text{Convex}(\mathcal{S})$ . For a PQ-tree  $T$ , the set of permutation  $\mathfrak{S}_{\text{PQ}}(T)$  is defined similarly as for FPQ-trees.

► **Lemma 8.** [24] *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  $\mathfrak{S}_{\text{PQ}}(T) = \text{Convex}(\mathcal{S})$  or decide that  $\text{Convex}(\mathcal{S}) = \emptyset$ .*





■ **Figure 6** An FPQ-tree  $T$  with  $\mathfrak{S}_{\text{FPQ}}(T) = \{abcd, cbad\}$ . The set of non-trivial common factors of  $\mathfrak{S}_T$  is  $\mathcal{I} = \{\{a, b, c\}, \{a, b\}, \{b, c\}\}$ . The permutations  $dabc$  and  $dcb a$  also belong to  $\text{Convex}(\mathcal{I})$ .

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  $\mathcal{C} = \langle \mathcal{N}_1, \dots, \mathcal{N}_k \rangle$  be a collection of nested sets  $\mathcal{N}_i \subseteq 2^X$  ( $1 \leq i \leq k$ ). Observe that a subset  $Y \subseteq X$  may occur in several nested sets of  $\mathcal{C}$ . We set  $\mathcal{S} = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$ . We say that a permutation  $\sigma \in \text{Convex}(\mathcal{S})$  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  $\text{Nested-Convex}(\mathcal{C}, \mathcal{S})$  denote the subset of permutations of  $\text{Convex}(\mathcal{S})$  that are  $\mathcal{C}$ -nested.

► **Lemma 9.** *Let  $\mathcal{C} = \langle \mathcal{N}_1, \dots, \mathcal{N}_k \rangle$  be a collection of nested sets such that for every  $1 \leq i \leq k$ ,  $\mathcal{N}_i \subset 2^X$ . If  $\text{Nested-Convex}(\mathcal{C}, \mathcal{S}) \neq \emptyset$ , with  $\mathcal{S} = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$ , then there exists an FPQ-tree  $T$  on  $X$  such that  $\mathfrak{S}_T = \text{Nested-Convex}(\mathcal{C}, \mathcal{S})$ . Moreover, such an FPQ-tree, when it exists, can be computed in polynomial time.*

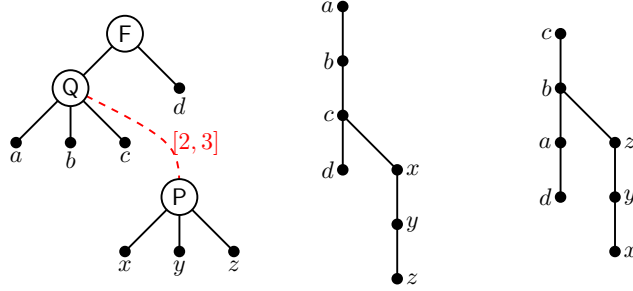
### 3.2 FPQ-hierarchies

A *hierarchy of ordered trees*  $H$  is defined on a set  $\mathcal{T} = \{T_0, T_1, \dots, T_p\}$  of non-trivial (i.e. with at least two vertices) 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 skeleton 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  $\ell_0$  the first leaf of  $\mathcal{L}(T_0)$  in  $\sigma_{T_0}$ . For every  $0 \leq i \leq p$ , the permutation  $\sigma_{T_i}$  of  $\mathcal{L}(T_i)$ , defined by  $T_i$ , is a path of  $T_H$ . Finally, for  $1 \leq i \leq p$ , let  $\ell_i$  be the first leaf of  $\mathcal{L}(T_i)$  in  $\sigma_{T_i}$ . 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  $\ell$  be the leaf of  $T_j$  that is a descendant of  $u_j$  and largest in  $\sigma_{T_j}$ . Then set  $\ell$  as the parent of  $\ell_i$  in  $T_H$ . See Figure 7 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  $H \approx_{\text{FPQ}} H'$ , if one can be turned into an FPQ-hierarchy isomorphic to the other by a series of  $\text{permute}(u)$  and  $\text{reverse}(u)$  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  $\text{reverse}(u)$  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  $\mathcal{T} = \{T_0, T_1, \dots, T_p\}$  of FPQ-trees defines a set  $\mathfrak{T}_{\text{FPQ}}(H)$  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  $\mathfrak{T}_{\text{FPQ}}(H)$  may not be isomorphic (see Figure 7).



■ **Figure 7** An FPQ-hierarchy  $H$ . The set  $\mathfrak{T}_{\text{FPQ}}(H)$  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, given a proper chordal graph  $G$  and a vertex  $x \in V$ , an FPQ-hierarchy  $H$  can be constructed to represent the set of indifference tree-layouts 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 combinatorics of the set of indifference tree-layouts rooted at  $x$ .

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 10.** *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  $(N(S) \cap V(C))$ -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. In the following paragraph we summarize the approach taken towards showing Theorem 11.

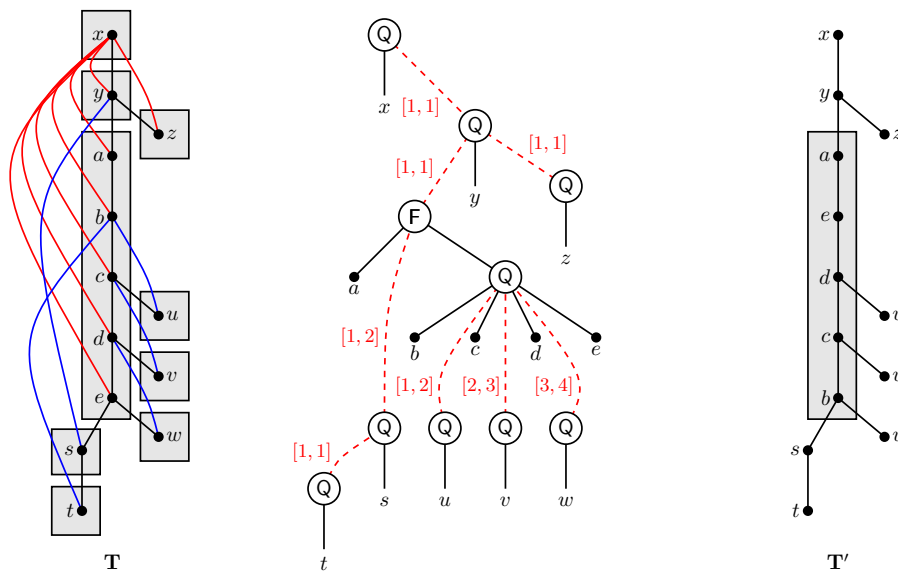
It can be shown that indifference tree-layouts are characterized as those such that for every vertex  $x$  distinct from  $\rho_G^{-1}(r)$ ,  $x$  is  $A_{\mathbf{T}_G}(x)$ -maximal and  $(N(A_{\mathbf{T}_G}(x)) \cap D_{\mathbf{T}_G}[x])$ -universal. This implies that every vertex  $x$  distinct from the root of an indifference tree-layout can be associated with a non-empty  $A_{\mathbf{T}_G}(x)$ -block containing  $x$ . Hereafter, we let  $B_{\mathbf{T}_G}(x)$  denote that block. If  $x = \rho_G^{-1}(r)$ , then we set  $B_{\mathbf{T}_G}(x) = \{x\}$ . Then, we prove that 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  $G$ . Moreover, we show that the set of  $\mathcal{B}(\mathbf{T}_G)$  containing the inclusion-maximal blocks of  $\mathbf{T}_G$ , partitions the vertex set of  $G$ . It follows that we can define the *block tree* of  $\mathbf{T}_G$ , which we denote  $\mathbf{T}_G|_{\mathcal{B}(\mathbf{T}_G)}$ , by contracting every block of  $\mathcal{B}(\mathbf{T}_G)$  into a single node. We get that if a connected proper chordal graph admits two indifference tree-layouts rooted at the same vertex, then the corresponding block trees are isomorphic,

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

Towards this, 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  $C_1, \dots, C_k$  be the connected components of  $G[C_{\mathbf{T}_G}(x) \setminus B_{\mathbf{T}_G}(x)]$ . We define  $\mathcal{C}_{\mathbf{T}_G}(x) = \langle \mathcal{N}_1, \dots, \mathcal{N}_k \rangle$  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  $\mathcal{S}_x = \bigcup_{1 \leq i \leq k} \mathcal{N}_i$ . We can prove that  $\mathcal{C}_{\mathbf{T}_G}(x) = \langle \mathcal{N}_1, \dots, \mathcal{N}_k \rangle$  is a collection of nested sets and that  $\text{Nested-Convex}(\mathcal{C}_{\mathbf{T}_G}(x), \mathcal{S}_x) \neq \emptyset$ .

The discussion above allows us to establish the existence of a canonical FPQ-hierarchy  $\mathbf{H}_G(x)$  encoding the set of indifference tree-layouts of  $G$  rooted at  $x$  (see Figure 8).

► **Theorem 11.** *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  $\mathbf{H}_G(x)$  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  $\mathbf{T}_G \in \mathfrak{T}_{\text{FPQ}}(\mathbf{H}_G(x))$ . Moreover  $\mathbf{H}_G(x)$  is unique and, given an indifference tree-layout rooted at  $x$ , it can be computed in polynomial time.*



■ **Figure 8** 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  $\mathbf{H}_G(x)$  is depicted in the middle. Observe that for the FPQ-tree  $\mathbf{T}$  of the block  $B_{\mathbf{T}}(a) = \{a, b, c, d, e\}$ ,  $\mathfrak{S}(\mathbf{T}) = \{abcde, aedcb\}$ . It follows that  $\mathfrak{T}_{\text{FPQ}}(\mathbf{H}_G(x))$  contains the two indifference tree-layouts  $\mathbf{T}$  and  $\mathbf{T}'$ .

## 5 Algorithmic aspects

In this section, we first design a polynomial time 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.

## 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  $x$ . We proceed in two steps. First, we compute the block tree  $\mathbf{B}_G^{\text{tree}}(x)$  of  $G$  rooted at  $x$  that would correspond to the skeleton tree of the FPQ-hierarchy  $\mathbf{H}_G(x)$  if  $G$  has an indifference tree-layout rooted at  $x$ . Then, in the second step, instead of computing  $\mathbf{H}_G(x)$ , we verify that  $\mathbf{B}_G^{\text{tree}}(x)$  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.** We assume that the input graph  $G = (V, E)$  is proper chordal and consider a vertex  $x \in V$  that is the root of some indifference tree-layout of  $G$ . As discussed above, the first step aims at computing the skeleton tree of  $\mathbf{H}_G(x)$  (see Theorem 11). That skeleton tree is the block tree  $\mathbf{B}_G^{\text{tree}}(x)$  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. To compute  $\mathbf{B}_G^{\text{tree}}(x)$ , 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  $x$ .

■ **Algorithm 1** Block tree computation.

---

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

```

1 set  $S \leftarrow \{x\}$ ,  $\mathcal{B} \leftarrow \{\{x\}\}$  and  $\mathbf{B}^{\text{tree}} \leftarrow (\mathcal{B}, \emptyset)$ ;
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  $\mathcal{B} \leftarrow \mathcal{B} \cup \{X\}$ ;
6     let  $B \in \mathcal{B}$  such that  $N(X) \cap B \neq \emptyset$  and that is the deepest;
7     add an edge in  $\mathbf{B}^{\text{tree}}$  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  $\mathbf{B}^{\text{tree}}$ ;
```

---

► **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  $\mathbf{B}_G^{\text{tree}}(x)$  that is the skeleton tree of the FPQ-hierarchy  $\mathbf{H}_G(x)$ .*

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

► **Observation 13.** Let  $x$  be a vertex of a graph  $G = (V, E)$ . If  $\mathbf{B}^{\text{tree}}$  is returned by Algorithm 1, then every extension  $T_{\mathbf{B}^{\text{tree}}}$  of  $\mathbf{B}^{\text{tree}}$  is a tree-layout of  $G$ . And moreover, if  $B, B',$  and  $B''$  are three blocks of  $\mathcal{B}$  such that  $B \prec_{\mathbf{B}^{\text{tree}}} B' \prec_{\mathbf{B}^{\text{tree}}} B''$ , then for every  $y \in B, y' \in B', y'' \in B'', yy'' \in E$  implies that  $y'y \in E$  and  $y'y'' \in E$ .

**Nested sets.** From Observation 13, an extension of  $\mathbf{B}^{\text{tree}}$  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,  $\mathbf{B}^{\text{tree}} = \mathbf{B}_G^{\text{tree}}(x)$ . It then follows from the proof of Theorem 11 that  $\mathbf{T}_G$  is an extension of  $\mathbf{B}^{\text{tree}}$ . The second step of the algorithm consists in testing if  $\mathbf{B}^{\text{tree}}$  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  $\mathbf{B}_G^{\text{tree}}(x)$ . To every block  $B$  of  $\mathcal{B}_G(x)$ , we assign a collection of nested subsets of  $2^B$  which we denote  $\mathcal{C}_B = \langle \mathcal{N}_1, \dots, \mathcal{N}_k \rangle$  (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  $B$ ,  $\text{Nested-Convex}(\mathcal{C}_B, \mathcal{S}_B) \neq \emptyset$  where  $\mathcal{S}_B = \cup_{1 \leq i \leq k} \mathcal{N}_i$ .

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

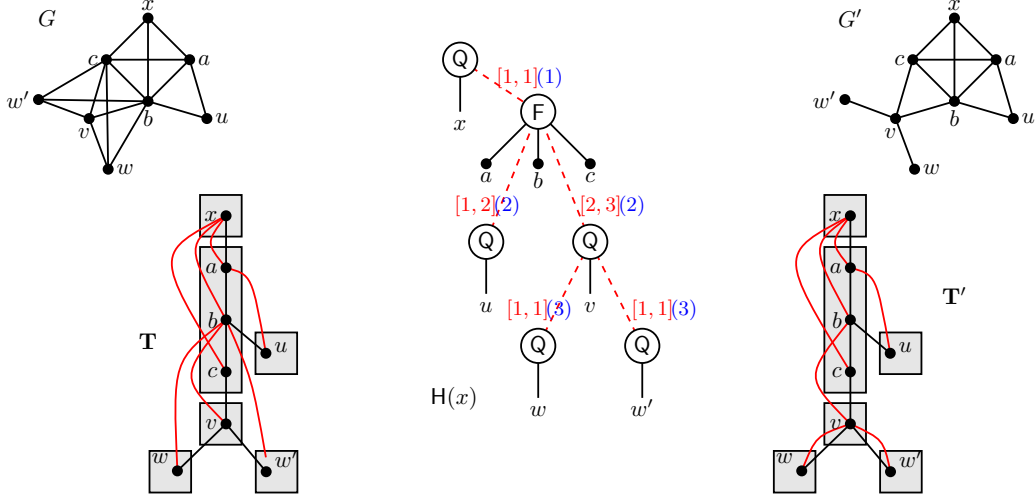
---

► **Theorem 14.** 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.

## 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 9) satisfying the conditions of Theorem 11. More precisely, given an FPQ-hierarchy of a proper chordal graph  $G$  that satisfies the conditions of Theorem 11, 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  $\mathbf{H}_G(x)$  since only the intersection of  $N(y)$  with the parent block is present

in  $H_G(x)$ . In the example of Figure 9, the vertices  $w$  and  $w'$  are also adjacent to  $c$  and  $b$ , which do not belong to their parent block.



■ **Figure 9** 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,  $H(x)$  is an FPQ-hierarchy such that  $\mathfrak{T}_{\text{FPQ}}(H(x))$  contains all the indifference tree-layouts rooted at  $x$  of  $G$  and of  $G'$ . We obtain  $H^*(x)$  for  $G$  by adding to  $H(x)$  the blue labels on the skeleton edges.

Let  $H_G(x)$  be the FPQ-hierarchy satisfying the conditions of Theorem 11, we define the *indifference FPQ-hierarchy*, denoted  $H_G^*(x)$ , obtained from  $H_G(x)$  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  $H_1^* \approx_{\text{FPQ}}^* H_2^*$ , if  $H_1 \approx_{\text{FPQ}} H_2$  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  $f : X_1 \rightarrow X_2$  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 15.** *Let  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  be two (connected) proper chordal graphs. Let  $H_1^*(x_1)$  be an indifference FPQ-hierarchy of  $G_1$  and  $H_2^*(x_2)$  be an indifference FPQ-hierarchy of  $G_2$ . Then  $H_1^*(x_1) \approx_{\text{FPQ}}^* H_2^*(x_2)$  if and only if  $G_1$  and  $G_2$  are isomorphic with  $x_1$  mapped to  $x_2$ .*

From Lemma 15, 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 [36]. That is, we adapt the standard unordered tree isomorphism algorithm that assigns to every unordered tree a canonical *isomorphism code* [47, 1]. Testing isomorphism then amounts to testing equality between two isomorphism codes.

We proceed with a detailed description of the isomorphism test. 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:

$$L <_{\text{lex}} F <_{\text{lex}} P <_{\text{lex}} Q <_{\text{lex}} 0 <_{\text{lex}} 1 \dots <_{\text{lex}} n <_{\text{lex}} \dots,$$

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

- if  $\text{type}(t) = F$ , then the identity permutation is the unique eligible permutation;
- if  $\text{type}(t) = P$ , then every permutation is eligible;
- if  $\text{type}(t) = Q$ , then the identity or its reverse permutation are the two eligible permutations.

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

$$\text{code}(t, \beta) = \begin{cases} \text{size}(t) \circ \text{type}(t) \circ \\ \text{code}(b_{\beta(1)}) \circ \dots \circ \text{code}(b_{\beta(k)}) \circ \\ \text{label}(s_{\pi_\beta(1)}, \beta) \circ \text{code}(s_{\pi_\beta(1)}) \circ \dots \circ \text{label}(s_{\pi_\beta(\ell)}, \beta) \circ \text{code}(s_{\pi_\beta(\ell)}) \end{cases}$$

where:

- $\text{type}(t) \in \{L, F, P, Q\}$ , indicates whether  $t$  a leaf (L), a F-node, a P-node, or a Q-node.
- $\text{size}(t) \in \mathbb{N}$ , stores the number of nodes in the sub-hierarchy rooted at  $t$  (including  $t$ ).
- $\text{label}(s, \beta)$ , with  $s$  being a skeleton child  $s$  of  $t$  and  $\beta$  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  $\text{label}(s) = \langle I^\beta(e), A(e) \rangle$ , where

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

- $\pi_\beta$  is, for some permutation  $\beta$  of  $[1, k]$ , a permutation of  $[1, \ell]$  that minimizes, with respect to  $<_{\text{lex}}$ :

$$\text{label}(s_{\pi_\beta(1)}, \beta) \circ \text{code}(s_{\pi_\beta(1)}) \circ \dots \circ \text{label}(s_{\pi_\beta(\ell)}, \beta) \circ \text{code}(s_{\pi_\beta(\ell)}).$$

Using the previous definitions, we can show that if  $H_1^*$  and  $H_2^*$  are indifference FPQ-hierarchies of the graphs  $G_1$  and  $G_2$  respectively, then  $H_1^* \approx_{\text{FPQ}}^* H_2^*$  if and only if  $\text{code}(H_1^*) = \text{code}(H_2^*)$ .

► **Theorem 16.** *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  $\mathbf{T}_1 \in \mathfrak{T}_{\text{FPQ}}(H_1^*)$ . This can be done in polynomial time by Theorem 14. 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  $H_1^* \approx_{\text{FPQ}}^* H_2^*$ . 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 15, if one of these tests is positive, then we can conclude that  $G_1$  and  $G_2$  are isomorphic graphs. ◀

## 6 Conclusion

A rough analysis of the complexity of the algorithms would lead to a  $O(n^4)$ -time complexity for the proper chordal graph recognition problem and a for the isomorphism test. We let open the question of deriving a faster algorithms. 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  $\mathbf{P}$  for proper chordal graphs shows that they form a non-trivial potential island of tractability for many other algorithmic problems. In this line, we leave open the status of Hamiltonian cycle, which is polynomial time solvable in proper interval graphs [4, 30] and interval graphs [31, 5], but NP-complete on strongly chordal graphs [38]. 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, and etc.), which are NP-complete by [33], 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 [13], Hell et al. [28] 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 A. Brandstädt, V.B. Le, and J. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.
- 9 S. Chaplick. Intersection graphs of non-crossing paths. In *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\_24.
- 10 M. Chein, M. Habib, and M.-C. Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37:35–50, 1981. doi:10.1016/0012-365X(81)90138-2.
- 11 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.
- 12 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.
- 13 Peter Damaschke. Forbidden ordered subgraphs. *Topics in Combinatorics and Graph Theory*, pages 219–229, 1990. doi:10.1007/978-3-642-46908-4\_25.
- 14 G. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961.
- 15 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.
- 16 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.
- 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 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.
- 21 M.C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978. doi:10.1016/0012-365X(78)90178-4.
- 22 M.C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980. doi:10.1016/C2013-0-10739-8.
- 23 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.
- 24 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.
- 25 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.
- 26 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.
- 27 G. Hajös. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11, 1957. Problem 65.

- 28 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.
- 29 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.
- 30 L. Ibarra. A simple algorithm to find hamiltonian cycles in proper interval graphs. *Information Processing Letters*, 109:1105–1108, 2009. doi:10.1016/j.ip1.2009.07.010.
- 31 J.M. Keil. Finding Hamiltonian circuits in interval graphs. *Information Processing Letters*, 20:201–206, 1985. doi:10.1016/0020-0190(85)90050-X.
- 32 H. Lerchs. On cliques and kernels. Technical report, Departement of Computer Science, University of Toronto, 1971.
- 33 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.
- 34 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.
- 35 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.
- 36 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.
- 37 R.H. Möhring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.
- 38 H. Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156:291–298, 1996. doi:10.1016/0012-365X(95)00057-4.
- 39 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.
- 40 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.
- 41 F.S. Roberts. *Representations of indifference relations*. PhD thesis, Stanford University, 1968.
- 42 F.S. Roberts. Indifference graphs. In F. Harrary, editor, *Proof Techniques in Graph Theory*, pages 139–146, 1969.
- 43 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.
- 44 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.
- 45 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.
- 46 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:10.1016/j.dam.2004.06.008.
- 47 G. Valiente. *Algorithms on trees and graphs*. Texts in Computer Science. Springer, 2002. doi:10.1007/978-3-030-81885-2.