Isomorphism Testing for T-graphs in FPT*

Deniz Ağaoğlu Çağırıcı[1][0000−0002−1691−0434] and
Petr Hliněný[2][0000−0003−2125−1514]

[1] Masaryk University, Brno, Czech Republic `agaoglu@mail.muni.cz`
[2] Masaryk University, Brno, Czech Republic `hlineny@fi.muni.cz`

Abstract. A T-graph (a special case of a chordal graph) is the intersection graph of connected subtrees of a suitable subdivision of a fixed tree T. We deal with the isomorphism problem for T-graphs which is *GI-complete* in general – when T is a part of the input and even a star. We prove that the T-graph isomorphism problem is in FPT when T is the fixed parameter of the problem. This can equivalently be stated that isomorphism is in FPT for chordal graphs of (so-called) bounded leafage. While the recognition problem for T-graphs is not known to be in FPT wrt. T, we do *not* need a T-representation to be given (a promise is enough). To obtain the result, we combine a suitable isomorphism-invariant decomposition of T-graphs with the classical tower-of-groups algorithm of Babai, and reuse some of the ideas of our isomorphism algorithm for S_d-graphs [MFCS 2020].

Keywords: chordal graph · H-graph · leafage · graph isomorphism · parameterized complexity.

1 Introduction

Two graphs G and H are called *isomorphic*, denoted by $G \simeq H$, if there is a bijection $f : V(G) \to V(H)$ such that for every pair $u, v \in V(G)$, $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$. The well-known *graph isomorphism problem* asks whether two input graphs are isomorphic, and it can be solved efficiently for various special graph classes [1, 9, 12, 14, 18, 23]. On the other hand, it is still unknown whether this problem is polynomial-time solvable or not (though, it is not expected to be NP-hard) in the general case, and a problem is said to be *GI-complete* if it is polynomial-time equivalent to the graph isomorphism.

We now briefly introduce two complexity classes of parameterized problems. Let k be the parameter, n be the input size, f and g be two computable functions, and c be some constant. A decision problem is in the *class FPT* (or *FPT-time*) if there exists an algorithm solving that problem correctly in time $\mathcal{O}(f(k) \cdot n^c)$. Similarly, a decision problem is in the *class XP* if there exists an algorithm solving that problem correctly in time $\mathcal{O}(f(k) \cdot n^{g(k)})$. Some parameters which yield to *FPT*- or *XP*-time algorithms for the graph isomorphism problem can be listed as tree-depth [10], tree-width [21], maximum degree [7] and genus [23]. In

---

this paper, we consider the parameterized complexity of the graph isomorphism problem for special instances of intersection graphs which we introduce next.

The *intersection graph* for a finite family of sets is an undirected graph G where each set is associated with a vertex of G, and each pair of vertices in G are joined by an edge if and only if the corresponding sets have a non-empty intersection. Chordal and interval graphs are two of the most well-known intersection graph classes related to our research.

A graph is *chordal* if every cycle of length more than three has a chord. They are also defined as the intersection graphs of subtrees of some (non-fixed) tree T [16]. Chordal graphs can be recognized in linear time, and they have linearly many maximal cliques which can be listed in polynomial time [24]. Deciding the isomorphism of chordal graphs is a *GI-complete* problem [25]. A graph G is an *interval graph* if it is the intersection graph of a set of intervals on the real line. Interval graphs form a subclass of chordal graphs. They can also be recognized in linear time, and interval graph isomorphism can be solved in linear time [9].

A subdivision of a graph G is the operation of replacing selected edge(s) of G by new induced paths (informally, putting new vertices to the middle of an edge). For a fixed graph H, an *H-graph* is the intersection graph of connected subgraphs of a suitable subdivision of the graph H [8], and they generalize many types of intersection graphs. For instance, interval graphs are K_2-graphs, their generalization called circular-arc graphs are K_3-graphs, and chordal graphs are the union of T-graphs where T ranges over all trees. We, however, consider *T-graphs* where T is a fixed tree. Even though chordal graphs can be recognized in linear time [24], deciding whether a given chordal graph is a T-graph is NP-complete when T is on the input [19]. In [11], Chaplick et al. gave an *XP*-time algorithm to recognize T-graphs parameterized by the size of T.

S_d-graphs form a subclass of T-graphs where S_d is the star with d rays. The isomorphism problem for S_d-graphs, and therefore for T-graphs, was shown to be GI-complete [25] with d on the input. In [4], we have proved by algebraic means that S_d-graph isomorphism can be solved in *FPT*-time parameterized by d, and then in [5] we have extended this approach to an *XP*-time algorithm for the isomorphism problem of T-graphs parameterized by the size of T. We have also considered in [5] the special case of isomorphism of proper T-graphs with a purely combinatorial *FPT*-time algorithm.

New contribution. In this paper, we show that the graph isomorphism problem for T-graphs can be solved in *FPT*-time parameterized by the size of T. Our algorithm does not assume or rely on T-representations of the input graphs to be given, and in fact it uses only some special properties of T-graphs.

Moreover, our result can be equivalently reformulated as an *FPT*-time algorithm for testing isomorphism of chordal graphs of bounded leafage, where the *leafage* of a chordal graph G can be defined as the least number of leaves of a tree T such that G is a T-graph. Since there is only a bounded number of trees T of a given number of leaves, modulo subdivisions, the correspondence of the two formulations is obvious.

Highly informally explaining our approach (which is different from [5]), we use chordality and properties of assumed T-representations of input graphs G and G' to efficiently compute their special hierarchical canonical decompositions into so-called fragments (Section 2). Each fragment will be an interval graph, and the isomorphism problem of interval graphs is well understood. Then we use some classical group-computing tools (Section 3, Babai's tower-of-groups approach) to compute possible "isomorphisms" between the decompositions of G and of G' (Section 4); each such isomorphism mapping between the fragments of the two decompositions, and simultaneously between the neighborhood sets of fragments in other fragments "higher up" in the decomposition.

We remark that the same problem has been independently and concurrently solved by Arvind, Nedela, Ponomarenko and Zeman [2],[3] using different means (by reducing the problem to automorphisms of colored order-3 hypergraphs with bounded sizes of color classes).

Statements marked with an asterisk (*) have proofs in the attached Appendix.

2 Structure and decomposition of T-graphs

In this section, we give a procedure to "extract" a bounded number of special interval subgraphs (called *fragments*) of a T-graph G in a way which is invariant under automorphisms and does not require a T-representation on input. Informally, the fragments can be seen as suitable "pieces" of G which are placed on the leaves of T in some representation, and their most important aspects are their simplicity and limited number. We use this extraction procedure repeatedly (and recursively) to obtain the full decomposition of a T-graph.

Structure of chordal graphs. We now give several useful terms and facts related to chordal graphs. A vertex v of a graph G is called *simplicial* if its neighborhood corresponds to a clique of G. It is known that every chordal graph contains a simplicial vertex and, by removing the simplicial vertices of a chordal graph repeatedly, one obtains an empty graph.

A *weighted clique graph* $\mathcal{C}_G$ of a graph G is the graph whose vertices are the maximal cliques of G and there is an edge between two vertices in $\mathcal{C}_G$ whenever the corresponding maximal cliques have a non-empty intersection. The edges in $\mathcal{C}_G$ are weighted by the cardinality of the intersection of the corresponding cliques.

A *clique tree* of G is any maximum-weight spanning tree of $\mathcal{C}_G$ which may not be unique. An edge of $\mathcal{C}_G$ is called *indispensable* (resp. *unnecessary*) if it appears in every (resp. none) maximum-weight spanning tree of $\mathcal{C}_G$. If G is chordal, every maximum-weight spanning tree T of $\mathcal{C}_G$ is also a T-representation of G, e.g. [22].

---

[3] To be completely accurate, our paper was first time submitted to a conference at the beginning of July 2021, and [2] appeared on arXiv just two weeks later, without mutual influence regarding the algorithms.

For a graph G and two vertices $u \neq v \in V(G)$, a subset $S \subseteq V(G)$ is called a *u-v separator* (or *u-v cut*) of G if u and v belong to different components of $G - S$. When $|S| = 1$, then S is called a *cutvertex*. S is called *minimal* if no proper subset of S is a u-v separator. Minimal separators of a graph are the separators which are minimal for some pair of vertices. Chordal graphs, thus T-graphs, have linearly many minimal vertex separators [17].

A *leaf clique* of a T-graph G is a maximal clique of G which can be a leaf of some clique tree of G (informally, it can be placed on a leaf of T in some T-representation of G). We use the following lemma in our algorithm:

Lemma 2.1 (Matsui et al. [22]). *A maximal clique C of a chordal graph G can be a leaf of a clique tree if and only if C satisfies (1) C is incident to at most one indispensable edge of $\mathcal{C}_G$, and (2) C is not a cutvertex in $\mathcal{C}'_G$ which is the subgraph of $\mathcal{C}_G$ which includes all edges except the unnecessary ones. The conditions can be checked in polynomial time.*

Decomposing T-graphs. The overall goal now is to recursively find a unique decomposition of a given T-graph G into levels such that each level consists of a bounded number of interval fragments.

For an illustration, a similar decomposition can be obtained directly from a T-representation of G: pick the interval subgraphs of G which are represented exclusively on the leaf edges of T, forming the outermost level, and recursively in the same way obtain the next levels. Unfortunately, this is not a suitable solution for us, not only that we do not have a T-decomposition at hand, but mainly because we need our decomposition to be *canonical*, meaning invariant under automorphisms of the graph, while this depends on a particular representation.

The contribution of this section is to compute such a decomposition the right canonical way. As sketched above, the core task is to canonically determine in the given graph G one bounded-size collection of fragments which will form the outermost level of the decomposition, and then the rest of the decomposition is obtained in the same way from recursively computed collections of fragments in the rest of the graph, which is also a T-graph[4].

For a chordal graph G and a (fixed) collection $Z_1, Z_2, \ldots, Z_s \subseteq G$ of distinct cliques, we write $Z_i \preceq Z_j$ if there exists $k \in \{1, \ldots, s\} \setminus \{i, j\}$ such that Z_j *separates* Z_i from Z_k in G (meaning that there is no path from $Z_i \setminus Z_j$ to $Z_k \setminus Z_j$ in $G - Z_j$), and say that $Z_i \preceq Z_j$ is witnessed by Z_k. Note that $\preceq$ is transitive, and hence a preorder. Let $Z_i \succneqq Z_j$ mean that $Z_i \preceq Z_j$ but $Z_j \npreceq Z_i$. We also write $Z_i \approx Z_j$ if there exists $k \in \{1, \ldots, s\} \setminus \{i, j\}$ such that both $Z_i \preceq Z_j$ and $Z_j \preceq Z_i$ hold and are witnessed by Z_k. Note that $Z_j \approx Z_i$ is stronger than just saying '$Z_i \preceq Z_j$ and $Z_j \preceq Z_i$,' and that $Z_i \cap Z_j$ then separates $Z_i \Delta Z_j$ from Z_k.

---

[4] Since the requirement of canonicity of our collection does not allow us to relate this collection to a particular T-representation of G, we cannot say whether the rest of G (after removing our collection of fragments) would be a T_1-graph for some strict subtree $T_1 \subsetneq T$, or only a T-graph again. That is why we speak about T-graphs for the same T (or, we could say graphs of bounded leafage here) throughout the whole recursion. In particular, we cannot directly use this procedure to recognize T-graphs.

Lemma 2.2. (*) *Let T be a tree with d leaves, and G be a T-graph. Assume that $Z_1, \ldots, Z_s \subseteq G$ are distinct cliques of G such that one of the following holds:*
a) for each $1 \leq i \leq s$, the set Z_i is a maximal clique in G, and for any $1 \leq i \neq j \leq s$, neither $Z_i \npreceq Z_j$ nor $Z_i \approx Z_j$ is true, or
b) for each $1 \leq i \leq s$, the set Z_i is a minimal separator in G cutting off a component F of $G - Z_i$ such that F contains a simplicial vertex of a leaf clique of G, and that F is disjoint from all Z_j, $j \neq i$.
Then $s \leq d$.

Let $Z \subseteq G$ be a minimal separator in G and $F \subseteq G$ a connected component of $G - Z$. Then Z is a clique since G is chordal, and whole Z is in the neighborhood of F by minimality. We call a *completion of F* (in implicit G) the graph F^+ obtained by contracting all vertices of G not in $V(F) \cup Z$ into one vertex l (the neighborhood of l is thus Z) and joining l with a new leaf vertex l', called the *tail of F^+*. Since F determines Z in a chordal graph G, the term F^+ is well defined.

We call a collection of disjoint nonempty induced subgraphs (not necessarily connected) $X_1, X_2, \ldots, X_s \subseteq G$, such that there are no edges between distinct X_i and X_j, a *fragment collection* of G of size s. We first give our procedure for computing a fragment collection, and subsequently formulate (and prove) the crucial properties of the computed collection and the whole decomposition.

Procedure 2.3 Let T be a tree with d leaves and no degree-2 vertex. Assume a T-graph G on the input. We compute an induced (and canonical) fragment collection $X_1, X_2, \ldots, X_s \subseteq G$ of G of size $0 < s \leq 2d$ as follows:

1. List all maximal cliques in G (using a simplicial decomposition) and compute the weighted clique graph $\mathcal{C}_G$ of G. Compute the list $\mathcal{L}$ of all possible leaf cliques of G by Lemma 2.1; in more detail, using [22, Algorithm 2] for computation of the indispensable edges in $\mathcal{C}_G$.
2. For every pair $L_1, L_2 \in \mathcal{L}$ such that $L_1 \preceq L_2$, remove L_2 from the list. Let $\mathcal{L}_0 \subseteq \mathcal{L}$ be the resulting list of cliques, which is nonempty since $\npreceq$ is acyclic.

3. Let $\mathcal{L}_1 := \{ L \in \mathcal{L}_0 : \forall L' \in \mathcal{L}_0 \setminus \{L\}.\ L \napprox L' \}$ be the subcollection of cliques incomparable with others in $\approx$. By Lemma 2.2(a) we have $|\mathcal{L}_1| \leq d$. If $\mathcal{L}_1 \neq \emptyset$, then **output** the following fragment collection of G: for each $L \in \mathcal{L}_1$, include in it the set $F \subseteq L$ of all simplicial vertices of L in the graph G.
4. Now, for each $L \in \mathcal{L}_0$ we have $L' \in \mathcal{L}_0 \setminus \{L\}$ such that $L \approx L'$ (and so $L \cap L'$ is a separator in G). For distinct $L_1, L_2 \in \mathcal{L}_0$ such that $L_1 \approx L_2$, we call a set $Z \subseteq L_1 \cap L_2$ a *joint separator for L_1, L_2* if Z separates $L_1 \triangle L_2$ from $L \setminus Z$ for some (any) $L \in \mathcal{L}_0 \setminus \{L_1, L_2\}$. We compute the family $\mathcal{Z}$ of all inclusion-minimal sets Z which are joint separators for some pair $L_1 \approx L_2 \in \mathcal{L}_0$ as above, over all such pairs L_1, L_2. This is efficient since all minimal separators in chordal graphs can be listed in linear time. Note that no set $Z \in \mathcal{Z}$ contains any simplicial vertex of G, and so $V(G) \nsubseteq \bigcup \mathcal{Z}$.
5. Let $\mathcal{C}$ be the family of the connected components of $G - \bigcup \mathcal{Z}$, and $\mathcal{C}_0 \subseteq \mathcal{C}$ consist of such $F \in \mathcal{C}$ that F is incident to just one set $Z_F \in \mathcal{Z}$. Note that

$\mathcal{C}_0 \neq \emptyset$, since otherwise the incidence graph between $\mathcal{C}$ and $\mathcal{Z}$ would have a cycle and this would in turn contradict chordality of G. Let $\mathcal{Z}_0 := \{Z_F \in \mathcal{Z} : F \in \mathcal{C}_0\}$. Moreover, by Lemma 2.2(b), $|\mathcal{Z}_0| \leq d$.

6. We make a collection $\mathcal{C}_0'$ from $\mathcal{C}_0$ by the following operation: for each $Z \in \mathcal{Z}_0$, take all $F \in \mathcal{C}_0$ such that $Z_F = Z$ and every vertex of F is adjacent to whole Z, and join them into one graph in $\mathcal{C}_0'$ (note that there can be arbitrarily many such F for one Z). Remaining graphs of $\mathcal{C}_0$ stay in $\mathcal{C}_0'$ without change. Then, we denote by $\mathcal{C}_1 \subseteq \mathcal{C}_0'$ the subcollection of those $F \in \mathcal{C}_0'$ such that the completion F^+ of F (in G) is an interval graph.[5]

7. If $\mathcal{C}_1 \neq \emptyset$, then **output** $\mathcal{C}_1$ as the fragment collection. (As we can show from Lemma 2.2(b), $|\mathcal{C}_1| \leq d + |\mathcal{Z}_0| \leq 2d$.)

8. Otherwise, for each graph $F \in \mathcal{C}_0'$, we call this procedure recursively on the completion F^+ of F (these calls are independent since the graphs in $\mathcal{C}_0'$ are pairwise disjoint). Among the fragments returned by this call, we keep only those which are subgraphs of F.[6] We **output** the fragment collection formed by the union of kept fragments from all recursive calls.

One call to Procedure 2.3 clearly takes only polynomial time (in some steps this depends on G being chordal – e.g., listing all cliques or separators). Since the possible recursive calls in the procedure are applied to pairwise disjoint parts of the graph (except the negligible completion of F to F^+), the overall computation of Procedure 2.3 takes polynomial time regardless of d. Regarding correctness, we are proving that $s \leq 2d$, which is in the respective Steps 3 and 7 indicated as a corollary of Lemma 2.2, except in the last (recursive) Step 8 where it can be derived in a similar way from Lemma 2.2 applied to the final collection. We leave the remaining technical details for the attached Appendix.

The last part is to prove a crucial fact that the collection $X_1, X_2, \ldots, X_s \subseteq G$ is indeed canonical, which is precisely stated as follows:

Lemma 2.4. (*) *Let G and G' be isomorphic T-graphs. If Procedure 2.3 computes the canonical collection $X_1, \ldots, X_s$ for G and the canonical collection $X_1', \ldots, X_{s'}'$ for G', then $s = s'$ and there is an isomorphism between G and G' matching in some order $X_1, \ldots, X_s$ to $X_1', \ldots, X_s'$.*

Levels, attachments and terminal sets. Following Procedure 2.3, we now show how the full decomposition of a T-graph G is completed.

For every fragment X of the canonical collection computed by Procedure 2.3, we define the list of *attachment sets* of X in $G - X$ as follows. If $X = F$ is obtained in Step 3, then it has one attachment set $L \setminus F$. Otherwise (Steps 6 and 7), the attachment sets of $X = F$ are all subsets A of the corresponding separator Z (of F) such that some vertex of X has the neighborhood in Z equal to A. Observe that the attachment sets of X are always cliques contained in the

---

[5] Informally, $F^+ \in \mathcal{C}_1$ iff F has an interval representation (on a horizontal line) to which its separator Z_F can be "attached from the left" on the same line.

[6] Note that, e.g., the separator and tail of F^+ may also be involved in a recursively computed fragment.

completion X^+, as defined above. Moreover, it is important that the attachment sets of X form a chain by the set inclusion, since G is chordal, and hence they are *uniquely determined* independently of automorphisms of X^+.

Procedure 2.5 Given a T-graph G, we determine a **canonical decomposition** of G recursively as follows. Start with $i = 1$ and $G_0 := G$.

1. Run Procedure 2.3 for G_{i-1}, obtaining the collection $X_1, \ldots, X_s$.
2. We call the special interval subgraphs $X_1, \ldots, X_s$ *fragments* and their family $\mathcal{X}_i := \{X_1, \ldots, X_s\}$ a *level* (of number i) of the constructed decomposition.
3. Let $G_i := G - \big(V(X_1) \cup \ldots \cup V(X_s)\big)$. Mark every attachment set of each X_j in G_i as a *terminal set*. These terminal sets will be further refined when recursively decomposing G_i; namely, further constructed fragments of G_i will inherit induced subsets of marked terminal sets as their terminal sets.
4. As long as G_i is not an interval graph, repeat this from Step 1 with $i \leftarrow i+1$.

Regarding this procedure, we stress that the obtained levels are numbered "from outside", meaning that the first (outermost) level is of the least index. The rule is that fragments from lower levels have their attachment sets as terminal sets in higher levels. As it will be made precise in the next section, an isomorphism between two T-graphs can be captured by a mapping between their canonical decompositions, which relates pairwise isomorphic fragments and preserves the incidence (i.e., identity) between the attachment sets of mapped fragments and the terminal sets of fragments in higher levels. See also Figure 1.

3   Group-computing tools

We first recall the notion of the *automorphism group* which is closely related to the graph isomorphism problem. An *automorphism* is an isomorphism of a graph G to itself, and the *automorphism group* of G is the group $Aut(G)$ of all automorphisms of G. There exists an isomorphism from G_1 to G_2 if and only if the automorphism group of the disjoint union $H := G_1 \uplus G_2$ contains a permutation exchanging the vertex sets of G_1 and G_2. We work with automorphism groups by means of their generators; a subset A of elements of a group Γ is called a *set of generators* if the members of A together with the operation of Γ can generate each element of Γ.

There are two related classical algebraic tools which we shall use in the next section. The first one is an algorithm performing computation of a subgroup of an arbitrary group, provided that we can efficiently test the membership in the subgroup and the subgroup is not "much smaller" than the original group:

Theorem 3.1. (Furst, Hopcroft and Luks [15, Cor. 1]) *Let Π be a permutation group given by its generators, and Π_1 be any subgroup of Π such that one can test in polynomial time whether $\pi \in \Pi_1$ for any $\pi \in \Pi$ (membership test). If the ratio $|\Pi|/|\Pi_1|$ is bounded by a function of a parameter d, then a set of generators of Π_1 can be computed in* FPT*-time (with respect to d).*

The second tool, known as Babai's "tower-of-groups" procedure (cf. [6]), will not be used as a standalone statement, but as a mean of approaching the task of computation of the automorphism group of our object H (e.g., graph). This procedure can be briefly outlined as follows; imagine an inclusion-ordered chain of groups $\Gamma_0 \supseteq \Gamma_1 \supseteq \ldots \supseteq \Gamma_{k-1} \supseteq \Gamma_k$ such that

- Γ_0 is a group of some unrestricted permutations on the ground set of our H,
- for each $i \in \{1, \ldots, k\}$, we "add" some further restriction (based on the structure of H) which has to be satisfied by all permutations of Γ_i,
- the restriction in the previous point is chosen such that the ratio $|\Gamma_{i-1}|/|\Gamma_i|$ is guaranteed to be "small", and
- in Γ_k, we get the automorphism group of our object H.

Then Theorem 3.1 can be used to compute Γ_1 from Γ_0, then Γ_2 from Γ_1, and so on until we get the automorphism group Γ_k.

Automorphism group of a decomposition. Here we are going to apply the above procedure in order to compute the automorphism group of a special object which combines the decompositions (cf. Procedure 2.5) of given T-graphs G_1 and G_2, but abstracts from precise structure of the fragments as interval graphs.

Consider canonical decompositions of the graphs G_1 and G_2, as produced by Procedure 2.5 in the form of level families $\mathcal{X}_1^1, \ldots, \mathcal{X}_\ell^1$ and $\mathcal{X}_1^2, \ldots, \mathcal{X}_{\ell'}^2$, respectively. We may assume that $\ell = \ell'$ since otherwise we immediately answer 'not isomorphic'. A combined decomposition of $H = G_1 \uplus G_2$ hence consists of the levels $\mathcal{X}_i := \mathcal{X}_i^1 \cup \mathcal{X}_i^2$ for $i = 1, \ldots, \ell$ and their respective terminal sets. More precisely, let $\mathcal{X} := \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_\ell$. Let $\mathcal{A}[X]$ for $X \in \mathcal{X}_k$ be the family of all terminal sets in X (as marked by Procedure 2.5 and then restricted to $V(X)$), and specially $\mathcal{A}^i[X] \subseteq \mathcal{A}[X]$ be those terminal sets in X which come from attachment sets of fragments on level $i < k$. Let $\mathcal{A}_k := \bigcup_{X \in \mathcal{X}_k} \mathcal{A}[X]$ and $\mathcal{A}_k^i := \bigcup_{X \in \mathcal{X}_k} \mathcal{A}^i[X]$ for $k = 1, \ldots, \ell$, and let $\mathcal{A} := \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_\ell$.

Recall, from Section 2, the definition of the completion X^+ of any $X \in \mathcal{X}_i$ which, in the current context, is defined with respect to the subgraph of H induced on the union U of vertex sets of $\mathcal{X}_{i+1} \cup \ldots \cup \mathcal{X}_\ell$ (of the higher levels from X). This is, exactly, the completion of X defined by the call to Procedure 2.3 on the level i which defined X as a fragment. Recall also the attachment sets of X which are subsets of U (in X^+) and invariant on automorphisms of X^+.

The *automorphism group of such a decomposition of H* (Figure 1) acts on the ground set $\mathcal{X} \cup \mathcal{A}$, and consists of permutations ϱ of $\mathcal{X} \cup \mathcal{A}$ which, in particular, map $\mathcal{X}_i$ onto $\mathcal{X}_i$ and $\mathcal{A}_i$ onto $\mathcal{A}_i$ for all $i = 1, \ldots, \ell$. Overall, we would like the permutation ϱ correspond to an actual automorphism of the graph H, for which purpose we introduce the following definition. A permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an *automorphism of the decomposition $(\mathcal{X}, \mathcal{A})$ of H* if the following hold true;

(A1) for each $X \in \mathcal{X}_i$ where $i \in \{1, \ldots, \ell\}$, we have $\varrho(X) \in \mathcal{X}_i$, and there is a graph isomorphism from the completion X^+ to the completion $\varrho(X)^+$ mapping the tail of X^+ to the tail of $\varrho(X)^+$ and the terminal sets in $\mathcal{A}^j[X]$ to the terminal sets in $\mathcal{A}^j[\varrho(X)]$ for each $1 \leq j < i$, and

Fig. 1: An illustration of a (combined) canonical decomposition of the graph $H = G_1 \uplus G_2$ into ℓ levels, with the collections of fragments $\mathcal{X}$ (thick black circles) and of terminal sets $\mathcal{A}$ (colored ellipses inside them). The arrows illustrate an automorphism of this decomposition: straight arrows show the possible mapping between isomorphic fragments on the same level, as in (A1), and wavy arrows indicate preservation of the incidence between attachment sets and the corresponding terminal sets, as stated by condition (A2).

(A2) for every $X \in \mathcal{X}_i$ and $A \in \mathcal{A}_k^i$ where $i \in \{1, \ldots, \ell\}$ and $k \in \{i+1, \ldots, \ell\}$, we have that if A is an attachment set of the fragment X (so, $A \subseteq X^+$), then $\varrho(A) \subseteq \varrho(X)^+$ is the corresponding attachment set of the fragment $\varrho(X)$.

Notice the role of the last two conditions. While (A1) speaks about consistency of ϱ with the actual graph H on the same level, (A2) on the other hand ensures consistency "between the levels". Right from this definition we get:

Proposition 3.2. (*) *Let $H = G_1 \uplus G_2$ and its canonical decomposition (Procedure 2.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ be as above. A permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an automorphism of this decomposition $(\mathcal{X}, \mathcal{A})$ of H, if and only if there exists a graph automorphism of H which acts on $\mathcal{X}$ and on $\mathcal{A}$ identically to ϱ.*

4   Main algorithm

We are now ready to present our main result which gives an *FPT*-time algorithm for isomorphism of T-graphs (without need for a given decomposition). The algorithm is based on Proposition 3.2, and so on efficient checking of the conditions (A1) and (A2) in the combined decomposition of two graphs. Stated precisely:

Theorem 4.1. *For a fixed tree T, there is an* FPT-*time algorithm that, given graphs G_1 and G_2, correctly decides whether $G_1 \simeq G_2$, or correctly answers that one or both of G_1 and G_2 are not T-graphs.*[7]

We first state a reformulation of it as a direct corollary.

Corollary 4.2. (*) *The graph isomorphism problem of chordal graphs G_1 and G_2 is in* FPT *parameterized by the leafage of G_1 and G_2.*

Theorem 4.1 now follows using Procedure 2.5, basic knowledge of automorphism groups and Proposition 3.2, and the following refined statement.

Theorem 4.3. (*) *Assume two T-graphs G_1 and G_2, and their combined canonical decomposition (Procedure 2.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ in ℓ levels, as in Section 3. Let $s = \max_{1 \leq i \leq \ell} |\mathcal{X}_i|$ be the maximum size of a level, and t be an upper bound on the maximum antichain size among the terminal set families $\mathcal{A}[X]$ over each $X \in \mathcal{X}$. Then the automorphism group of the decomposition, defined by (A1) and (A2) above, can be computed in* FPT-*time with the parameter $s+t$.*

Notice that, in our situation, the parameter $s + t$ indeed is bounded in terms of $|T|$; we have $s \leq 2d$ and $t \leq d$ directly from the arguments in Lemma 2.2 and Procedure 2.3. Due to space limits, we give only a sketch of the proof here.

Proof (sketch). First, we outline that the condition (A1) can be dealt with (in Step 1 below) efficiently w.r.t. the parameter t: the arguments combine the known and nice description of interval graphs via so-called PQ-trees [9,13], with an *FPT*-time algorithm [4] for the automorphism group of set families with bounded-size antichain (where the latter assumption is crucial for this to work).

Using the previous, we prove the rest as a commented algorithm outline:

1. For every level $k \in \{1, \ldots, \ell\}$ of the decomposition of $H = G_1 \uplus G_2$ we compute the following permutation group Λ_k acting on $\mathcal{X}_k \cup \mathcal{A}_k$.

   a) We partition $\mathcal{X}_k$ into classes according to the isomorphism condition (A1); i.e., $X_1, X_2 \in \mathcal{X}_k$ fall into the same class iff there is a graph isomorphism from X_1^+ to X_2^+ preserving the tail and bijectively mapping $\mathcal{A}^i[X_1]$ to $\mathcal{A}^i[X_2]$ for all $1 \leq i < k$. We add the bounded-order symmetric subgroup on each such class of $\mathcal{X}_k$ to Λ_k.

   b) Now, for every permutation $\varrho \in \Lambda_k$ of $\mathcal{X}_k$ and all $X \in \mathcal{X}_k$, and for any chosen isomorphism $\iota_X : X^+ \to \varrho(X)^+$ conforming to (A1), we add to Λ_k the permutation of $\mathcal{A}_k$ naturally composed of partial mappings of the terminal sets induced by the isomorphisms ι_X over $X \in \mathcal{X}_k$.

   c) For every $X \in \mathcal{X}_k$, we compute generators of the automorphism subgroup of X^+ (preserving the tail) which maps $\mathcal{A}^i[X]$ to $\mathcal{A}^i[X]$ for every $1 \leq i < k$, and we add to Λ_k the action of each such generator on $\mathcal{A}[X] \subseteq \mathcal{A}_k$ (as a new generator of Λ_k). This part together with (a), as outlined above, is a nontrivial algorithmic task [3] and we provide further details in the attached Appendix.

---

[7] The latter outcome ('not a T-graph') happens when some of the assertions assuming a T-graph in Procedure 2.3 fails.

2. We let $\Gamma_0 = \Lambda_1 \times \ldots \times \Lambda_\ell$ be the direct product of the previous subgroups. Notice that Γ_0 is formed by the permutations conforming to condition (A1).

3. Finally, we apply Babai's tower-of-groups procedure [6] to Γ_0 in order to compute the desired automorphism group of the decomposition. We loop over all pairs $1 \leq i < j \leq \ell$ of levels and over all cardinalities r of terminal sets in $\mathcal{A}_j$, which is $\mathcal{O}(n^3)$ iterations, and in iteration $k = 1, 2 \ldots$ compute:

   * $\Gamma_k \subseteq \Gamma_{k-1}$ consisting of exactly those automorphisms which conform to the condition (A2) for every component $X \in \mathcal{X}_i$ and every terminal set $A \in \mathcal{A}_j^i$ such that $|A| = r$. Then Γ_k forms a subgroup of Γ_{k-1} (i.e., closed on a composition) thanks to the condition (A1) being true in Γ_{k-1}, and so we can compute Γ_k using Theorem 3.1.

4. We output the group Γ_m of the last iteration $k = m$ of Step 3 as the result.

Correctness of the outcome of this algorithm is self-explanatory from the outline; Γ_m satisfies (A1) and (A2) for all possible choice of X and A.

We finish with a brief argument of why the computation in Step 3 via Theorem 3.1 is indeed efficient. Observe that for all i, j, $|\mathcal{X}_i| \leq s$ and the number of $A \in \mathcal{A}_j^i$ such that $|A| = r$ is at most st. By standard algebraic means (counting cosets of Γ_k in Γ_{k-1}), we get that $|\Gamma_{k-1}|/|\Gamma_k|$ is bounded from above by the order of the subgroup "induced" on $\mathcal{X}_i$ times the order of the subgroup on considered sets A of cardinality r. The latter number is at most $s! \cdot (st)!$ regardless of Γ_{k-1}, and hence bounded in the parameter. $\qquad\square$

5   Conclusions

We have provided an FPT-time algorithm to solve the isomorphism problem for T-graphs with a fixed parameter $|T|$ and for chordal graphs of bounded leafage. There seems to be little hope to further extend this result for more general classes of chordal graphs since already for split graphs of unbounded leafage the isomorphism problem is GI-complete. Though, we may combine our result with that of Krawczyk [20] for circular-arc graphs isomorphism to possibly tackle the case of H-graphs for which H contains exactly one cycle.

On the other hand, an open question remains whether a similar decomposition technique as that in Section 2 can be used to solve the recognition problem of T-graphs in FPT-time, since the currently best algorithm [11] works in XP-time.

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974)
2. Arvind, V., Nedela, R., Ponomarenko, I., Zeman, P.: Testing isomorphism of chordal graphs of bounded leafage is fixed-parameter tractable. CoRR **abs/2107.10689** (2021), https://arxiv.org/abs/2107.10689

3. Ağaoğlu, D., Hliněný, P.: Automorphism group of marked interval graphs in FPT. CoRR **abs/2202.12664** (2022), `https://arxiv.org/abs/2202.12664`

4. Ağaoğlu, D., Hliněný, P.: Isomorphism problem for S_d-graphs. In: Esparza, J., Král', D. (eds.) 45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic. LIPIcs, vol. 170, pp. 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.MFCS.2020.4, `https://doi.org/10.4230/LIPIcs.MFCS.2020.4`

5. Ağaoğlu, D., Hliněný, P.: Efficient isomorphism for S_d-graphs and T-graphs. CoRR **abs/1907.01495** (2021)

6. Babai, L.: Monte Carlo algorithms in graph isomorphism testing. Tech. Rep. 79-10, Université de Montréal (1979), 42 pages

7. Babai, L., Luks, E.M.: Canonical labeling of graphs. In: Johnson, D.S., Fagin, R., Fredman, M.L., Harel, D., Karp, R.M., Lynch, N.A., Papadimitriou, C.H., Rivest, R.L., Ruzzo, W.L., Seiferas, J.I. (eds.) Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA. pp. 171–183. ACM (1983). https://doi.org/10.1145/800061.808746, `https://doi.org/10.1145/800061.808746`

8. Biró, M., Hujter, M., Tuza, Z.: Precoloring extension. i. interval graphs. Discrete Mathematics **100** pp. 267–279 (1992)

9. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**(3), 335–379 (1976). https://doi.org/10.1016/S0022-0000(76)80045-1, `https://doi.org/10.1016/S0022-0000(76)80045-1`

10. Bouland, A., Dawar, A., Kopczyński, E.: On tractable parameterizations of graph isomorphism. In: Thilikos, D.M., Woeginger, G.J. (eds.) Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7535, pp. 218–230. Springer (2012). https://doi.org/10.1007/978-3-642-33293-7_21, `https://doi.org/10.1007/978-3-642-33293-7_21`

11. Chaplick, S., Töpfer, M., Voborník, J., Zeman, P.: On H-topological intersection graphs. In: Bodlaender, H.L., Woeginger, G.J. (eds.) Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10520, pp. 167–179. Springer (2017). https://doi.org/10.1007/978-3-319-68705-6_13, `https://doi.org/10.1007/978-3-319-68705-6_13`

12. Colbourn, C.J.: On testing isomorphism of permutation graphs. Networks **11**(1), 13–21 (1981). https://doi.org/10.1002/net.3230110103, `https://doi.org/10.1002/net.3230110103`

13. Colbourn, C.J., Booth, K.S.: Linear time automorphism algorithms for trees, interval graphs, and planar graphs. SIAM J. Comput. **10**(1), 203–225 (1981). https://doi.org/10.1137/0210015, `https://doi.org/10.1137/0210015`

14. Curtis, A.R., Lin, M.C., McConnell, R.M., Nussbaum, Y., Soulignac, F.J., Spinrad, J.P., Szwarcfiter, J.L.: Isomorphism of graph classes related to the circular-ones property. Discret. Math. Theor. Comput. Sci. **15**(1), 157–182 (2013), `http://dmtcs.episciences.org/625`

15. Furst, M.L., Hopcroft, J.E., Luks, E.M.: Polynomial-time algorithms for permutation groups. In: 21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980. pp. 36–

41. IEEE Computer Society (1980). https://doi.org/10.1109/SFCS.1980.34, `https://doi.org/10.1109/SFCS.1980.34`

16. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. Journal of Combinatorial Theory, Series B **16**(1), 47–56 (1974). https://doi.org/https://doi.org/10.1016/0095-8956(74)90094-X, `https://www.sciencedirect.com/science/article/pii/009589567490094X`

17. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57). North-Holland Publishing Co., NLD (2004)

18. Hopcroft, J.E., Wong, J.K.: Linear time algorithm for isomorphism of planar graphs (preliminary report). In: Constable, R.L., Ritchie, R.W., Carlyle, J.W., Harrison, M.A. (eds.) Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA. pp. 172–184. ACM (1974). https://doi.org/10.1145/800119.803896, `https://doi.org/10.1145/800119.803896`

19. Klavík, P., Kratochvíl, J., Otachi, Y., Saitoh, T.: Extending partial representations of subclasses of chordal graphs. Theor. Comput. Sci. **576**, 85–101 (2015). https://doi.org/10.1016/j.tcs.2015.02.007, `https://doi.org/10.1016/j.tcs.2015.02.007`

20. Krawczyk, T.: Testing isomorphism of circular-arc graphs - Hsu's approach revisited. CoRR **abs/1904.04501** (2019), `http://arxiv.org/abs/1904.04501`

21. Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. SIAM J. Comput. **46**(1), 161–189 (2017). https://doi.org/10.1137/140999980, `https://doi.org/10.1137/140999980`

22. Matsui, Y., Uehara, R., Uno, T.: Enumeration of the perfect sequences of a chordal graph. Theor. Comput. Sci. **411**(40-42), 3635–3641 (2010). https://doi.org/10.1016/j.tcs.2010.06.007, `https://doi.org/10.1016/j.tcs.2010.06.007`

23. Miller, G.L.: Isomorphism testing for graphs of bounded genus. In: Miller, R.E., Ginsburg, S., Burkhard, W.A., Lipton, R.J. (eds.) Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA. pp. 225–235. ACM (1980). https://doi.org/10.1145/800141.804670, `https://doi.org/10.1145/800141.804670`

24. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. **5**(2), 266–283 (1976). https://doi.org/10.1137/0205021, `https://doi.org/10.1137/0205021`

25. Zemlyachenko, V.N., Korneenko, N.M., Tyshkevich, R.I.: Graph isomorphism problem. J. of Soviet Mathematics **29**, 1426–1481 (1985)

A   Additions to Section 2

The task here is to comment and add missing arguments to the computation of a canonical decomposition of a T-graph. We start with an illustration of a T-representation of a graph in Figure 2.

Fig. 2: A T-graph G shown by its T-representation (where T is drawn in thick dashed black lines, and the subtrees shown in colored solid lines represent the vertices of G). The picture also illustrates the first three (outermost) levels – in order magenta, green, blue, of the canonical decomposition of G obtained using Procedure 2.3. The rest of G is undistinguished in orange color. Notice that the magenta and green levels have been obtained from Step 3 of the procedure, while the blue level has come from Step 7.

The following observation will be useful in the coming proofs.

(*) If T is a tree with d leaves, and we mark $d+1$ vertices of T, then some path in T contains 3 marked vertices.

This can be proved by successively removing unmarked leaves from T, until all $\leq d$ leaves of the remaining subtree of T are marked. However, then also some internal vertex is marked, and it can be prolonged into a path ending in two leaves which are marked, too.

Lemma 2.2. (*) *Let T be a tree with d leaves, and G be a T-graph. Assume that $Z_1, \ldots, Z_s \subseteq G$ are distinct cliques of G such that one of the following holds:*
a) for each $1 \leq i \leq s$, the set Z_i is a maximal clique in G, and for any $1 \leq i \neq j \leq s$, neither $Z_i \gneqq Z_j$ nor $Z_i \approx Z_j$ is true, or
b) for each $1 \leq i \leq s$, the set Z_i is a minimal separator in G cutting off a component F of $G - Z_i$ such that F contains a simplicial vertex of a leaf clique of G, and that F is disjoint from all Z_j, $j \neq i$.
Then $s \leq d$.

Proof. Consider a T-representation of the graph G – as the intersection graph of subtrees of a subdivision T' of T. Then every clique X of G must be represented such that the representatives of all vertices of X intersect in (at least) one common node $v[X] \in V(T')$.

a) If $s > d$, then by observation (*) there exist $1 \leq i < j < k \leq s$ such that the three (distinct) nodes $v[Z_i]$, $v[Z_j]$ and $v[Z_k]$ of T' lie on one path in this order. Hence Z_j separates Z_i from Z_k in G by basic properties of a T-representation and maximality of the clique Z_j. If Z_i separates Z_j from Z_k, too, then $Z_i \approx Z_j$. Otherwise, we have $Z_i \not\succeq Z_j$, and both cases lead to a contradiction.

b) For $1 \leq i \leq s$ and some component F of $G - Z_i$, let X_i be a leaf clique of G having a simplicial vertex in F. As in a), assuming $s > d$, we find nodes $v[X_i]$, $v[X_j]$ and $v[X_k]$ of T' (of three distinct maximal leaf cliques X_i, X_j, X_k of G) that lie on one path in this order. However, from the definition of a T-representation (note that the node $v[X_j]$ disconnects $v[X_i]$ from $v[X_k]$ in T'), we get that $Z_i \cap Z_k \subseteq Z_j$. From the assumption of minimality of our separators we conclude that $Z_i = Z_j = Z_k$.   □

Additions to Procedure 2.3. We further comment on necessity of considering the joint separators in Step 4 and subsequent steps of the procedure, which is the more complicated side of the procedure. This is, though, unavoidable in cases that G contains arbitrarily many leaf cliques which are then mutually comparable in $\approx$. Such a situation is illustrated in Figure 3.

We also add a bit of explanation to Step 7: $|\mathcal{C}_1| \leq d + |\mathcal{Z}_0|$. The part $\leq |\mathcal{Z}_0|$ applies to those joined members of $\mathcal{C}_1$ which are fully adjacent to whole $Z \in \mathcal{Z}_0$, i.e., as $|\mathcal{C}_1 \setminus \mathcal{C}_0| \leq |\mathcal{Z}_0|$. The part $\leq d$ applies to the remaining members of $\mathcal{C}_1$, precisely, as $|\mathcal{C}_1 \cap \mathcal{C}_0| \leq |\mathcal{Z}_0|$ which is an application of Lemma 2.2(b). An analogous reasoning applies to the fragments collected from recursive calls in Step 8; these satisfy the same properties with respect to the whole T-representation of G as fragments directly obtained in Step 7.   □

Lemma 2.4. (*) *Let G and G' be isomorphic T-graphs. If Procedure 2.3 computes the canonical collection $X_1, \ldots, X_s$ for G and the canonical collection $X'_1, \ldots, X'_{s'}$ for G', then $s = s'$ and there is an isomorphism between G and G' matching in some order $X_1, \ldots, X_s$ to $X'_1, \ldots, X'_s$.*

Proof. One may easily verify that every step of Procedure 2.3 takes into account only isomorphism-invariant properties of the graph G, does not consider the input representation of G in any way and makes no arbitrary decisions. Consequently, every step performed by the procedure for the input G has an "isomorphic" step preformed for the input G'. This extends to possible recursive calls as well, and the conclusion follows.   □

B   Additions to Section 3

For further details regarding automorphism groups, see e.g., [15]. Here we briefly illustrate Babai's "tower-of-groups" procedure on the concrete example of Babai [6]

Fig. 3: A fragment of a T-representation of a graph G, where the branch to the right is formed from a leaf edge of T. Notice that this branch carries many maximal cliques (actually 7, but we can easily build many more there) which all except one can be leaf cliques, and so Procedure 2.3 finds their minimal separator consisting of the blue vertex and outputs the whole interval subgraph as one fragment.

(to which our use in Section 4 is conceptually very similar): A *d-bounded color multiplicity graph* is a graph G whose vertex set is arbitrarily partitioned into m color classes $V(G) = V_1 \cup \ldots \cup V_m$ such that $V_i \cap V_j = \emptyset$ for all $1 \leq i < j \leq m$. The number m of colors is arbitrary, but for all $1 \leq i \leq m$, the cardinality $|V_i|$, called the multiplicity of V_i, is at most d. To compute the automorphism group of such G, we start with Γ_0 which freely permutes each color class of G (formally, it is the product of the symmetric groups on each V_i). Then, stepwise, we add the restrictions to preserve the edges between V_i and V_j for $(i, j) = (1, 2), (1, 3), \ldots, (1, m), (2, 3), \ldots, (m - 1, m)$. The last group Γ_k for $k = \binom{m}{2}$ is the automorphism group of G and the total runtime is in *FPT* with the parameter d.

We return in a closer detail to the crucial correspondence between automorphisms of T-graphs and automorphisms of our special decompositions as defined in Section 3. Recall that a permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an *automorphism of the decomposition* of H if the following hold true;

(A1) for each $X \in \mathcal{X}_i$ where $i \in \{1, \ldots, \ell\}$, we have $\varrho(X) \in \mathcal{X}_i$, and there is a graph isomorphism from the completion X^+ to the completion $\varrho(X)^+$ mapping the tail of X^+ to the tail of $\varrho(X)^+$ and the terminal sets in $\mathcal{A}^j[X]$ to the terminal sets in $\mathcal{A}^j[\varrho(X)]$ for each $1 \leq j < i$, and

(A2) for every $X \in \mathcal{X}_i$ and $A \in \mathcal{A}_k^i$ where $i \in \{1, \ldots, \ell\}$ and $k \in \{i+1, \ldots, \ell\}$, we have that if A is an attachment set of the fragment X (so, $A \subseteq X^+$), then $\varrho(A) \subseteq \varrho(X)^+$ is the corresponding attachment set of the fragment $\varrho(X)$.

In regard of (A2) we remark that the words 'corresponding attachment set' refer to the fact that attachment sets of X are uniquely determined in the graph isomorphism to $\varrho(X)$.

Proposition 3.2. (*) *Let $H = G_1 \uplus G_2$ and its canonical decomposition (Procedure 2.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ be as above. A permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an automorphism of this decomposition $(\mathcal{X}, \mathcal{A})$ of H, if and only if there exists a graph automorphism of H which acts on $\mathcal{X}$ and on $\mathcal{A}$ identically to ϱ.*

Proof. In the 'only if' direction, we consider a permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ satisfying conditions (A1) and (A2), i.e., an automorphism ϱ of the considered decomposition of H. We take the mapping π on $V(H)$ which is composed of all isomorphism bijections from $X \in \mathcal{X}$ to $\varrho(X)$ claimed by (A1). Then π indeed is a permutation of $V(H)$ since ϱ is a permutation on $\mathcal{X}$, and π respects all edges of $E(H)$ which belong to some $X \in \mathcal{X}$. All remaining edges of H are between some fragment $X \in \mathcal{X}_i$ and one of its attachments ("higher up" in the decomposition) which coincides with some terminal set $A \in \mathcal{A}_j$ where $j > i$, by the way we decomposed H. Condition (A2) ensures that those edges of H are preserved as well by π, and hence π is an automorphism of H. See Figure 1.

In the 'if' direction, by recursive application of Lemma 2.4, we get that

- any automorphism of G_1 (or of G_2, up to symmetry) preserves the fragments and the levels of the decomposition of G_1 and, consequently, it preserves also the terminal sets by their incidence with attachments of the fragments;
- if $G_1 \simeq G_2$, we have an isomorphism $\iota : G_1 \to G_2$ preserving the fragments and the levels between G_1 and G_2, and then ι can be composed with any automorphism of G_2 from the previous point.

Consequently, for every graph automorphism σ of H we get an induced permutation on $\mathcal{X} \cup \mathcal{A}$, which indeed is an automorphism of the decomposition according to the conditions (A1) and (A2) – simply because σ was a graph automorphism. $\square$

C   Automorphisms of set families (or of hypergraphs)

In order to efficiently compute with terminal sets introduced by Procedure 2.5, we give the following technical result from [4].

Let $\mathcal{U}$ and $\mathcal{U}'$ be set families over finite ground sets Z and Z' (with no additional structure), respectively. A bijection π from $\mathcal{U}$ to $\mathcal{U}'$ is called an *isomorphism* if and only if there exists a related bijection ζ from Z to Z' such that π and ζ together preserve the incidence relation $\in$, i.e., for all $U \in \mathcal{U}$ and $z \in Z$ we have $z \in U \iff \zeta(z) \in \pi(U)$. This is essentially the same concept as that of an isomorphism between hypergraphs $(Z, \mathcal{U})$ and $(Z, \mathcal{U}')$, but notice that we primarily focus on the mapping between the sets of $\mathcal{U}$ and $\mathcal{U}'$, and not on the mapping between the elements of Z and Z'. An isomorphism π from $\mathcal{U}$ to $\mathcal{U}$ is an *automorphism of $\mathcal{U}$*.

For any set family $\mathcal{U}$, we call a *cardinality Venn diagram* of $\mathcal{U}$ the vector $\left(\ell_{\mathcal{U}, \mathcal{U}_1} : \emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U} \right)$ such that $\ell_{\mathcal{U}, \mathcal{U}_1} := |L_{\mathcal{U}, \mathcal{U}_1}|$ where $L_{\mathcal{U}, \mathcal{U}_1} = \bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$. That is, we record the cardinality of every internal cell of the Venn diagram of $\mathcal{U}$. Let $\pi(\mathcal{U}_1) = \{\pi(A) : A \in \mathcal{U}_1\}$ for $\mathcal{U}_1 \subseteq \mathcal{U}$.

The following is a straightforward but crucial observation:

Proposition C.1. *For a set family $\mathcal{U}$ over Z, a permutation π of $\mathcal{U}$ is an automorphism of $\mathcal{U}$ if and only if the cardinality Venn diagrams of $\mathcal{U}$ and of $\pi(\mathcal{U})$ are the same, meaning that $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\pi(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.*
The latter condition can be tested in time $\mathcal{O}(m^2)$ where $m = |\mathcal{U}| + |Z|$.

Notice that the problem of computing the automorphism group of such set family $\mathcal{U}$ is GI-complete in general – we can take $\mathcal{U}$ as the set of edges of a graph as 2-element subsets. Nevertheless, we can compute the group efficiently in the special case of our terminal sets which have bounded-size antichains:

Proposition C.2. ([5, Algorithm 2 and Lemma 5.9]) *Let $\mathcal{U}$ be a set family over a finite ground set Z such that the maximum size of an antichain in $\mathcal{U}$ is d (i.e., there are no more than d sets in $\mathcal{U}$ pairwise incomparable by the inclusion). Then the automorphism group of $\mathcal{U}$ can be computed in FPT-time with respect to d.*

To give a brief sketch of a proof here, we observe the following [4, Lemma 5.7]: If a permutation π on $\mathcal{U}$ fails to be an automorphism of $\mathcal{U}$, then there is a subfamily $\mathcal{U}_2 \subseteq \mathcal{U}$ witnessing this failure such that $\mathcal{U}_2$ is an antichain in the inclusion. Together with the bound $|\mathcal{U}_2| \leq d$ on antichains, this is enough to make Babai's tower-of-groups machinery work in *FPT*-time.

D   Automorphisms of Interval Graphs with Marked Sets

The task here is to argue that the algorithm given in Theorem 4.3 computes efficiently in Step 1, items (a) and (c), before finishing the full proof in the next section. Recall that the task is to compute the automorphism group of an interval graph G (which is easy in the basic setting [13]), but under an additional constraint that a given set family $\mathcal{A} \subseteq 2^{V(G)}$ (recall the terminal sets) is preserved by the automorphisms – that each set from $\mathcal{A}$ is mapped into a set from $\mathcal{A}$.

The latter problem is generally GI-hard since G may be chosen as a clique and $\mathcal{A}$ as the edge set of an arbitrary graph H, but the crucial restriction in our case is that the maximum size of an antichain of sets in $\mathcal{A}$ is bounded (cf. Proposition C.2). Then we obtain:

Lemma D.1. *Let G be an interval graph and $m > 0$ an integer. Let $\mathcal{A}^1, \ldots, \mathcal{A}^m$ be families (in general multisets) of subsets of $V(G)$ (terminal or marked sets of G) such that, for $\mathcal{A} := \mathcal{A}^1 \cup \ldots \cup \mathcal{A}^m$,*

- *every set $A \in \mathcal{A}$ induces a clique of G, and*
- *the maximum size of an antichain in $\mathcal{A}$ equals t (i.e., there are no more than t sets in $\mathcal{A}$ pairwise incomparable by the inclusion).*

Denote by Γ_1 the group consisting of those automorphisms σ of G such that, for each $i \in \{1, \ldots, m\}$, σ preserves the set family $\mathcal{A}^i$. Then one can in FPT-time with the parameter t (but independently of m) compute the group Γ of

permutations of $\mathcal{A}$ which is the action of Γ_1 on $\mathcal{A}$. In more detail, a permutation τ of $\mathcal{A}$ belongs to Γ, if and only if there exists an automorphism ϱ of G such that, for every $i \in \{1, \ldots, m\}$ and all $A \in \mathcal{A}^i$, we have $\tau(A) \in \mathcal{A}^i$ and $\tau(A) = \varrho(A)$.

Proof (sketch). All interval representations of an interval graph, or equivalently all its clique paths (recall the clique trees of chordal graphs from Section 2), can be represented by one suitable data structure called the *PQ-tree* [9]. A PQ-tree T is a rooted ordered tree whose internal nodes are labelled as either P-nodes or Q-nodes, where the children of a P-node can be arbitrarily reordered, while the order of the children of a Q-node can only be reversed. The leaves of T hold maximal cliques of our interval graph G. The following fact is crucial [9,13]:

(*) For every interval graph G one can in linear time construct a PQ-tree T (with leaves in a bijection with the maximal cliques of G), such that the permissible reorderings of T are in a one-to-one correspondence with all interval representations of G.

In particular, this means that every automorphism of G can be represented as a permissible reordering of T (though, not the other way round).

Every node p of a PQ-tree T of G can be associated with a subgraph of G formed by the union of all cliques of the descendant leaves of p – this subgraph *belongs* to p. Then for a node p we define the *inner vertices of p* as those vertices of G which belong to p and to at least two child nodes of p, but they do not belong to any sibling node of p. (In the case of a P-node, the inner vertices of p belong to all child nodes of p, but this is generally not true for Q-nodes.)

The first step is to reduce the tree T into a small subtree which is "essential" for the sets of $\mathcal{A}$. Precisely, call a node p of T *clean* if the inner vertices of p are disjoint from $\bigcup \mathcal{A}$. The subtree rooted at p is then *clean* if p and all descendants of p in T are clean. Observe that the number of non-clean subtrees at the same depth of T is always bounded from above by t. Indeed, since every set $A \in \mathcal{A}$ induces a clique in G, at most one of the considered non-clean subtrees can be caused by the same set A, and the witnessing sets of the non-clean subtrees form an antichain. Now, for every node q of T, we use [13] to determine the isomorphism class of the subgraph belonging to the PQ-tree formed by q and its clean subtrees. We store this information as an annotation of q and discard the clean subtrees. Let the resulting reduced tree be denoted by $T' \subseteq T$.

We show that the automorphisms of the reduced tree T' (with the aforementioned annotation) can be handled using the tools from Section C. By (*), we can in a canonical (i.e., automorphism-invariant) way decompose the vertex set of G into layers; where layer i is formed by the inner vertices of the nodes of T which are at depth i. In the subsequent argument, we show that structure of the tree T' can now be "replaced" by suitably chosen sets added to the terminal set family $\mathcal{A}$. For each node q of T' we, essentially, add the set B_q formed by the vertices of the subgraph of G belonging to q in T'. This does not increase the maximum antichain size, since B_q in an incomparable subfamily can be replaced by any set of $\mathcal{A}$ contained in B_q. We also keep the annotation of q as an annotation of the set B_q. Moreover, for a Q-node q, we annotate the order of

the children of q (in a reversible way). Finally, we can use Proposition C.2 to compute the (annotation-preserving) automorphism group of the resulting set family, which coincides with the desired permutation group Γ when restricted onto $\mathcal{A}$.

Since the previous claims are interesting on their own, we leave the full detailed description for a separate paper [3]. □

Corollary D.2. *Let, for $j = 1, 2$, G_j be a connected interval graph, $m > 0$ an integer, t a parameter, and $\mathcal{A}_j^1, \dots, \mathcal{A}_j^m$ families of subsets of $V(G_j)$, all as in Lemma D.1 for each $j \in \{1, 2\}$. Then one can in FPT-time with the parameter t decide whether there exists an isomorphism from G_1 to G_2 bijectively mapping $\mathcal{A}_1^i$ to $\mathcal{A}_2^i$ for all $1 \le i \le m$.*

Proof. We simply consider the interval graph G formed as the disjoint union of G_1 and G_2, compute the respective permutation group on $\mathcal{A}_1 \cup \mathcal{A}_2$ exactly as in Lemma D.1 and check whether some permutation exchanges $\mathcal{A}_1$ with $\mathcal{A}_2$. □

We, moreover, remark that the condition in Step 1 of Theorem 4.3 – namely that we require the isomorphisms / automorphisms to preserve the tail of X^+, can easily be respected in Lemma D.1 by introducing a separate family of a single set with the tail(s).

E   Additions to Section 4

We finish with the skipped detailed arguments for Section 4.

Corollary 4.2. (*) *The graph isomorphism problem of chordal graphs G_1 and G_2 is in FPT parameterized by the leafage of G_1 and G_2.*

Proof. Let the promised leafage of G_1 and G_2 be at most d. We exhaustively try all trees $T_1, T_2, \dots$ without degree-2 vertices and with at most d leaves; their total number depends only on d (exponentially). For each such T_i sequentially, we call the algorithm of Theorem 4.1 with $T = T_i$, and if we ever get an answer about $G_1 \simeq G_2$, we output it and quit. If all answers are that G_1 or/and G_2 are not T_i-graphs, then the promise of leafage $\le d$ is violated. □

Theorem 4.3. (*) *Assume two T-graphs G_1 and G_2, and their combined canonical decomposition (Procedure 2.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ in ℓ levels, as in Section 3. Let $s = \max_{1 \le i \le \ell} |\mathcal{X}_i|$ be the maximum size of a level, and t be an upper bound on the maximum antichain size among the terminal set families $\mathcal{A}[X]$ over each $X \in \mathcal{X}$. Then the automorphism group of the decomposition, defined by (A1) and (A2) above, can be computed in FPT-time with the parameter $s + t$.*

Proof. We refer to the algorithm outline in the main paper. Correctness of Steps 1 and 2 with respect to the condition (A1) is self-evident. Regarding efficiency of computation in Step 1 we refer to Lemma D.1 and Corollary D.2. The rest of these steps follows by standard computation with groups (which are represented by their sets of generators, as usually).

Correctness of Step 3 is again self-evident – we stepwise ensure that the resulting subgroup Γ_m satisfies by all its members the condition (A2), which together with aforementioned (A1) imply that Γ_m indeed is the automorphism group of the given decomposition of H.

We now add more details to justification of proper and efficient use of Theorem 3.1 is Step 3. In particular, this meas to show that the ratio $|\Gamma_{k-1}|/|\Gamma_k|$ is bounded in the parameters s, t in order to claim runtime in FPT. For the latter, consider any two permutations $\varrho, \sigma \in \Gamma_{k-1}$ which mutually agree on mapping of all fragments and terminal sets considered by Step 3 in the current iteration k. Hence the composed permutation $\sigma^{-1} \circ \varrho$ is identical on the elements currently considered by Step 3, and the condition (A2) in the current iteration k is automatically true for $\sigma^{-1} \circ \varrho$, meaning that $\sigma^{-1} \circ \varrho \in \Gamma_k$. Consequently, such ϱ and σ belong to the same coset of the subgroup Γ_k in Γ_{k-1} by the definition. It is well known that the number of these cosets equals $|\Gamma_{k-1}|/|\Gamma_k|$ which we would like to estimate.

Now, how many permutations in Γ_{k-1} are there that pairwise disagree on mapping of all components and terminal sets considered by Step 3? In iteration k of Step 3 we have at most s components to be mapped on level i', at most t terminal sets of cardinality r in every fragment (since these sets form an antichain). This gives a possibility of at most $s! \cdot (st)!$ distinct mappings, and hence $|\Gamma_{k-1}|/|\Gamma_k| \leq s! \cdot (st)!$ as needed by Theorem 3.1.   □

Finally, regarding the overall FPT-runtime of the whole algorithm composed of Procedures 2.3, 2.5, and Lemma D.1 and Theorem 3.1. The first two procedures run in polynomial time regardless of our parameters, and the latter two algorithms take each FPT-time with respect to parameters which are bounded by functions of the "master" parameter equal to the number of leaves of our fixed tree T.