Isomorphism Testing for $T$-graphs in FPT

Deniz Ağaoğlu$^1$ and Petr Hliněný$^2$

$^1$ Masaryk University, Brno, Czech Republic
$^2$ Masaryk University, Brno, Czech Republic

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, 8, 11, 13, 17, 22]. 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 $O(f(k) \cdot n^c)$. Similarly, a decision problem is in the class XP if there exists an algorithm solving that problem correctly in time $O(f(k) \cdot n^{g(k)})$. Some parameters which yield to FPT- or XP-time algorithms for the graph isomorphism problem can be listed as tree-depth [9], tree-width [20], maximum degree [6] and genus [22]. In

* This work was supported by the Czech Science Foundation, project no. 20-04567S.
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 \( T \) [15]. Chordal graphs can be recognized in linear time, and they have linearly many maximal cliques which can be listed in polynomial time [23]. Deciding the isomorphism of chordal graphs is a GI-complete problem [24]. 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 [8].

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 \( H \) [7], 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 [18]. In [10], 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 [24] with \( d \) on the input. In [3], we have proved by algebraic means that \( S_d \)-graph isomorphism can be solved in FPT-time parameterized by \( d \), and then in [4] 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 [4] 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.
Isomorphism Testing for T-graphs in FPT

Highly informally explaining our approach (which is different from [4]), 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] using different means (by reducing the problem to automorphisms of colored order-3 hypergraphs with bounded sizes of color classes).

Due to restricted space, statements marked with an asterisk (*) have proofs only in the full paper.

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 CG of a graph G is the graph whose vertices are the maximal cliques of G and there is an edge between two vertices in CG whenever the corresponding maximal cliques have a non-empty intersection. The edges in CG are weighted by the cardinality of the intersection of the corresponding cliques.

A clique tree of G is any maximum-weight spanning tree of CG which may not be unique. An edge of CG is called indispensable (resp. unnecessary) if it appears in every (resp. none) maximum-weight spanning tree of CG. If G is chordal, every maximum-weight spanning tree T of CG is also a T-representation of G, e.g. [21].

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 \textit{u-v separator} (or \textit{u-v cut}) of $G$ if $u$ and $v$ belong to different components of $G - S$. When $|S| = 1$, then $S$ is called a \textit{cutvertex}. $S$ is called \textit{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 [10].

A \textit{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:

\begin{lemma}[Matsui et al. [21]] 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 $C_G$, and (2) $C$ is not a cutvertex in $C_G$ which is the subgraph of $C_G$ which includes all edges except the unnecessary ones. The conditions can be checked in polynomial time.
\end{lemma}

\textbf{Decomposing \textit{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 \textit{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 $Z_i \not\preceq Z_j$ mean that $Z_i \preceq Z_j$ but $Z_j \not\preceq Z_i$. We also write $Z_i \simeq Z_j$ 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 \textquoteleft{$Z_i \preceq Z_j$ and $Z_j \preceq Z_i$'} and that $Z_i \cap Z_j$ then separates $Z_i \Delta Z_j$ from $Z_k$.

\begin{footnote}{1} 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 $T_1 \subseteq T$, 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.
\end{footnote}
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 any $1 \leq i \neq j \leq s$, neither $Z_i \not\subset Z_j$ 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 $s \leq d$.

If $G$ is a chordal graph, $Z \subseteq G$ a minimal separator in $G$ and $F \subseteq G$ a connected component of $G - Z$. Note that $Z$ is a clique since $G$ is chordal, and that 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 $C_G$ of $G$. Compute the list $L$ of all possible leaf cliques of $G$ by Lemma 2.4 in more detail, using [21 Algorithm 2] for computation of the indispensable edges in $C_G$.

2. For every pair $L_1, L_2 \in L$ such that $L_1 \not\subseteq L_2$, remove $L_2$ from the list. Let $\mathcal{L}_0 \subseteq L$ be the resulting list of cliques, which is nonempty since $\not\subseteq$ is acyclic.

3. Let $\mathcal{L}_1 := \{L \in \mathcal{L}_0 : \forall L' \in \mathcal{L}_0 \setminus \{L\}, L \neq L'\}$ be the subcollection of cliques incomparable with others in $\mathcal{L}$. 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 $L_1 \Delta L_2$ from $L \setminus Z$ for some (any) $L \in \mathcal{L}_0 \setminus \{L_1, L_2\}$. We compute the family $Z$ 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 $Z \in Z$ contains any simplicial vertex of $G$, and so $V(G) \not\subseteq \bigcup Z$.

5. Let $\mathcal{C}$ be the family of the connected components of $G - \bigcup Z$, and $C_0 \subseteq \mathcal{C}$ consist of such $F \in \mathcal{C}$ that $F$ is incident to just one set $Z_F \in Z$. Note that
6. A˘gao˘glu and P. Hlin˘en´y

\[ C_0 \neq \emptyset, \text{ since otherwise the incidence graph between } C \text{ and } Z \text{ would have a cycle and this would in turn contradict chordality of } G. \text{ Let } Z_0 := \{ Z_F \in Z : F \in C_0 \}. \text{ Moreover, by Lemma 2.2(b), } |Z_0| \leq d. \]

6. We make a collection \( C'_0 \) from \( C_0 \) by the following operation: for each \( Z \in Z_0 \), take all \( F \in C_0 \) such that \( Z_F = Z \) and every vertex of \( F \) is adjacent to whole \( Z \), and join them into one graph in \( C'_0 \) (note that there can be arbitrarily many such \( F \) for one \( Z \)). Remaining graphs of \( C_0 \) stay in \( C'_0 \) without change. Then, we denote by \( C_1 \subseteq C'_0 \) the subcollection of those \( F \in C'_0 \) such that the completion \( F^+ \) of \( F \) (in \( G \)) is an interval graph.  

7. If \( C_1 \neq \emptyset \), then \textbf{output} \( C_1 \) as the fragment collection. (As we can show from Lemma 2.2(b), \( |C_1| \leq d + |Z_0| \leq 2d \).) 

8. Otherwise, for each graph \( F \in C'_0 \), we call this procedure recursively on the completion \( F^+ \) of \( F \) (these calls are independent since the graphs in \( C'_0 \) are pairwise disjoint). Among the fragments returned by this call, we keep only those which are subgraphs of \( F \).  

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 must prove that \( s \leq 2d \), which follows from stated Lemma 2.2 except in the last (recursive) step where it can be derived in a way similar (albeit more complicated) to Lemma 2.2. We leave the remaining technical details for the full paper.

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:

\textbf{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 \( X'_1, \ldots, X'_{s'} \).

\textbf{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 \textit{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 completion

\[ F^+ \subseteq C_1 \text{ iff } F \text{ has an interval representation (on a horizontal line) to which its separator } Z_F \text{ can be “attached from the left” on the same line.} \]

\[ \text{Note that, e.g., the separator and tail of } F^+ \text{ may also be involved in a recursively computed fragment.} \]
Isomorphism Testing for T-graphs in FPT

is 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 \( X_i := \{ X_1, \ldots, X_s \} \) a level (of number \( i \)) of the constructed decomposition.
3. Let \( G_i := G - (V(X_1) \cup \ldots \cup V(X_s)) \). 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 \( \text{Aut}(G) \) 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 \( G_2 \). In fact, assuming connectivity of the graphs \( G_1 \) and \( G_2 \), it is enough to look for a permutation mapping some vertex of \( G_1 \) to a vertex of \( G_2 \), and only among generators of the automorphism group. Recall that a subset \( A \) of elements of a group \( \Gamma \) is called a set of generators if the members of \( A \) together with the operation of \( \Gamma \) can generate each element of \( \Gamma \).

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 [14, Cor. 1]) Let \( \Pi \) be a permutation group given by its generators, and \( \Pi_1 \) be any subgroup of \( \Pi \) 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 \( \Pi_1 \) can be computed in FPT-time (with respect to \( d \)).
The second tool, known as Babai’s “tower-of-groups” procedure (cf. [5]), 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

- \( \Gamma_0 \) 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 \( \Gamma_i \),
- the restriction in the previous point is chosen such that the ratio \( |\Gamma_{i-1}|/|\Gamma_i| \) is guaranteed to be “small”, and
- in \( \Gamma_k \), we get the automorphism group of our object \( H \).

Then Theorem 3.1 can be used to compute \( \Gamma_1 \) from \( \Gamma_0 \), then \( \Gamma_2 \) from \( \Gamma_1 \), and so on until we get the automorphism group \( \Gamma_k \).

**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 graphs.

Consider canonical decompositions of the graphs \( G_1 \) and \( G_2 \), as produced by Procedure 2.5 in the form of level families \( \mathcal{X}_1^1, \ldots, \mathcal{X}_1^\ell \) and \( \mathcal{X}_2^1, \ldots, \mathcal{X}_2^\ell \), 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 \( \mathcal{A}_k := \bigcup_{X \in \mathcal{A}_k} \mathcal{A}[X] \) and \( \mathcal{A}_k^i := \bigcup_{X \in \mathcal{A}_k^i} \mathcal{A}_i[X] \) for \( k = 1, \ldots, \ell \), and let \( \mathcal{A}_i := \mathcal{A}_1^i \cup \ldots \cup \mathcal{A}_\ell^i \).

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 \( \varrho \) 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 \( \varrho \) correspond to an actual automorphism of the graph \( H \), for which purpose we introduce the following definition. A permutation \( \varrho \) of \( \mathcal{X} \cup \mathcal{A} \) is an automorphism of the decomposition of \( H \) if the following hold true:

1. (A1) for each \( X \in \mathcal{X}_i \) \( (i \in \{1, \ldots, \ell\}) \), 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 \( \mathcal{A}[X] \) to the terminal sets in \( \mathcal{A}[\varrho(X)] \) for each \( 1 \leq j < i \), and
Fig. 1: An illustration of a (combined) canonical decomposition of the graph $H = G_1 \sqcup G_2$ into $\ell$ 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 $A \in \mathcal{A}^i_k$ 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 $\varphi(A) \subseteq \varphi(X)^+$ is the corresponding attachment set of the fragment $\varphi(X)$.

Notice the role of the last two conditions. While (A1) speaks about consistency of $\varphi$ 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 $H = G_1 \sqcup G_2$ and its canonical decomposition (Procedure 2.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ be as above. A permutation $\varphi$ of $\mathcal{X} \cup \mathcal{A}$ is an automorphism of this decomposition, if and only if there exists a graph automorphism of $H$ which acts on $\mathcal{X}$ and on $\mathcal{A}$ identically to $\varphi$.

### 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.

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 decomposition (Procedure 2.5) formed by families $X$ and $A$ in $\ell$ levels, as in Section 3. Let $s = \max_{1 \leq i \leq \ell} |X_i|$ be the maximum size of a level, and $t$ be an upper bound on the maximum antichain size among the terminal set families $A^i[X]$ over each $X \in X$. 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 Procedure 2.3.

Due to space limits, we give only a sketch of proof in this short paper.

Proof (sketch). First, we state that condition (A1) can be dealt with (as below) efficiently w.r.t. the parameter $t$. Briefly, the arguments combine known and very nice description of interval graphs via so-called PQ-trees [8, 12], with an FPT-time algorithm [3] 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 $A_k$ acting on $X_k \cup A_k$.
   a) We partition $X_k$ into classes according to the isomorphism condition (A1), i.e., $X_1, X_2 \in X_k$ fall into the same class iff there is a graph isomorphism from $X^+_1$ to $X^+_2$ preserving the tail and bijectively mapping $A^i[X_1]$ to $A^i[X_2]$ for all $1 \leq i < k$. We add the bounded-order symmetric subgroup on each such class of $X_k$ to $A_k$.
   b) Now, for every permutation $\varrho \in A_k$ of $X_k$ and all $X \in X_k$, and for any chosen isomorphism $\iota_X : X^+ \to \varrho(X)^+$ conforming to (A1), we add to $A_k$ the permutation of $A_k$ naturally composed of partial mappings of the terminal sets induced by the isomorphisms $\iota_X$ over $X \in X_k$.
   c) For every $X \in X_k$, we compute generators of the automorphism subgroup of $X^+$ mapping $A^i[X]$ to $A^i[X]$ for all $1 \leq i < k$, and we add to $A_k$ the action of each such generator on $A[X] \subseteq A_k$ (as a new generator of $A_k$). This is a nontrivial algorithmic task and we provide the details in the full paper.

\[\text{The latter outcome (‘not a }T\text{-graph’)} \text{ happens when some of the assertions assuming a } T\text{-graph in Procedure 2.3 fails.}\]
2. We let $\Gamma_0 = A_1 \times \ldots \times A_\ell$ be the direct product of the previous subgroups. Notice that $\Gamma_0$ is formed by the permutations conforming to condition (A1).

3. Finally, we apply Babai’s tower-of-groups procedure [5] to $\Gamma_0$ 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 $A_j$, which is $O(n^3)$ 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 $X \in X_i$ and every terminal set $A \in A_i$ such that $|A| = r$. Then $\Gamma_k$ forms a subgroup of $\Gamma_{k-1}$ (i.e., closed on a composition) thanks to the condition (A1) being true in $\Gamma_{k-1}$, and so we can compute $\Gamma_k$ using Theorem 3.1.

4. We output the last group $\Gamma_m$ of step 3 as the result.

Correctness of the outcome of this algorithm is self-explanatory from the outline; $\Gamma_m$ 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 $i, j, |X_i| \leq s$ and the number of $A \in A_i$ such that $|A| = r$ is at most $st$. By standard algebraic means (counting cosets of $\Gamma_k$ in $\Gamma_{k-1}$), we get that $|\Gamma_{k-1}|/|\Gamma_k|$ is bounded from above by the order of the subgroup “induced” on $X_i$ times the order of the subgroup on considered sets $A$ of cardinality $r$. The latter number is at most $s! \cdot (st)!$ regardless of $\Gamma_{k-1}$, and hence bounded in the parameter.

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 [19] 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 [10] works in XP-time.

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The design and analysis of computer algorithms. Addison–Wesley, Reading, Mass. 1974 (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)
3. Ağaoğlu, D., Hliněný, P.: Isomorphism problem for $S_d$-graphs. In: MFCS 2020. LIPIcs, vol. 170, pp. 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

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

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

6. Babai, L., Luks, E.M.: Canonical labeling of graphs. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing. p. 171–183. STOC ’83 (1983)

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

8. 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, 335–379. (1976)

9. Bouland, A., Dawar, A., Kopczyński, E.: On tractable parameterizations of graph isomorphism. In: IPEC 2012. Lecture Notes in Computer Science, vol. 7535, pp. 218–230. Springer (2012)

10. Chaplick, S., Töpfer, M., Voborník, J., Zeman, P.: On H-topological intersection graphs. CoRR abs/1608.02389 (2016)

11. Colbourn, C.J.: On testing isomorphism of permutation graphs. Networks 11(1), 13–21 (1981)

12. 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)

13. Curtis, A.R., Lin, M., McConnell, R., Nussbaum, Y., Soulignac, F.J., Spinrad, J., Szewczyk, J.: Isomorphism of graph classes related to the circular-ones property. Discret. Math. Theor. Comput. Sci. 15, 157–182 (2013)

14. Furst, M.L., Hopcroft, J.E., Luks, E.M.: Polynomial-time algorithms for permutation groups. 21st Annual Symposium on Foundations of Computer Science (FOCS) pp. 36–41 (1980)

15. 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)

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

17. Hopcroft, J.E., Wong, J.K.: Linear time algorithm for isomorphism of planar graphs. In: STOC. pp. 172–184 (1974)

18. Klavík, P., Kratochvíl, J., Otachi, Y., Saitho, T.: Extending partial representations of subclasses of chordal graphs. Theoretical Computer Science 576, 85 – 101 (2015)

19. Krawczyk, T.: Testing isomorphism of circular-arc graphs - Hsu’s approach revisited. CoRR abs/1904.04501 (2019)

20. Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Fixed-parameter tractable canonicalization and isomorphism test for graphs of bounded treewidth. SIAM Journal on Computing pp. 161–189 (2017)

21. Matsui, Y., Uehara, R., Uno, T.: Enumeration of the perfect sequences of a chordal graph. Theoretical Computer Science 411(40), 3635–3641 (2010)

22. Miller, G.: Isomorphism testing for graphs of bounded genus. In: Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing. pp. 225–235. STOC ’80 (1980)

23. Rose, D., Lueker, G., Tarjan, R.E.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal on Computing, 5 (2): 266–283 (1976)
24. Zemlyachenko, V.N., Korneenko, N.M., Tyshkevich, R.I.: Graph isomorphism problem. Journal of Soviet Mathematics 29 pp. 1426–1481 (1985)