Recognizing Proper Tree-Graphs

Steven Chaplick
Maastricht University, The Netherlands
s.chaplick@maastrichtuniversity.nl

Petr A. Golovach
Department of Informatics, University of Bergen, Norway
petr.golovach@uib.no

Tim A. Hartmann
RWTH Aachen, Germany
hartmann@algo.rwth-aachen.de

Dušan Knop
Department of Theoretical Computer Science, Faculty of Information Technology,
Czech Technical University in Prague, Czech Republic
dusan.knop@fit.cvut.cz

Abstract

We investigate the parameterized complexity of the recognition problem for the proper H-graphs. The H-graphs are the intersection graphs of connected subgraphs of a subdivision of a multigraph H, and the properness means that the containment relationship between the representations of the vertices is forbidden. The class of H-graphs was introduced as a natural (parameterized) generalization of interval and circular-arc graphs by Biró, Hujter, and Tuza in 1992, and the proper H-graphs were introduced by Chaplick et al. in WADS 2019 as a generalization of proper interval and circular-arc graphs. For these graph classes, H may be seen as a structural parameter reflecting the distance of a graph to a (proper) interval graph, and as such gained attention as a structural parameter in the design of efficient algorithms. We show the following results.

- For a tree T with t nodes, it can be decided in $2^{\mathcal{O}(t^2 \log t)} \cdot n^3$ time, whether an n-vertex graph G is a proper T-graph. For yes-instances, our algorithm outputs a proper T-representation. This proves that the recognition problem for proper H-graphs, where H required to be a tree, is fixed-parameter tractable when parameterized by the size of T. Previously only NP-completeness was known.
- Contrasting to the first result, we prove that if H is not constrained to be a tree, then the recognition problem becomes much harder. Namely, we show that there is a multigraph H with 4 vertices and 5 edges such that it is NP-complete to decide whether G is a proper H-graph.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Fixed parameter tractability

Keywords and phrases intersection graphs, H-graphs, recognition, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.IPEC.2020.8

Related Version `https://arxiv.org/abs/2011.11670`

Funding *Petr A. Golovach*: Author supported by the project MULTIVAL of the Research Council of Norway.
Dušan Knop: Supported by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16 019/0000765 "Research Center for Informatics".

Acknowledgements We thank Peter Zeman for initial discussions on this work, particularly relating to the NP-completeness of proper H-graph recognition. We also thank an anonymous reviewer to point to a mistake in an earlier version.

© Steven Chaplick, Petr A. Golovach, Tim A. Hartmann, and Dušan Knop;
licensed under Creative Commons License CC-BY

1 Introduction

An *intersection representation* of a graph $G = (V, E)$ is a collection of nonempty sets $\{M_v \mid v \in V(G)\}$ over a given universe such that $\{u, v\}$ is an edge of G if and only if $M_u \cap M_v \neq \emptyset$. A large area of research in graph algorithms is the study of restricted families of graphs arising from specialized intersection representations, e.g., the *interval* graphs are the graphs with an intersection representation where the sets are intervals of $\mathbb{R}$, and the *circular-arc* graphs are intersection graphs of families of arcs of the circle. The interval graphs and similarly defined graph classes are often motivated from application areas such as circuit layout problems [24, 4], scheduling problems [22], biological problems [19], or the study of wireless networks [16]. We refer to the books [5, 15] for an introduction and survey of the known results on the related graph classes.

A key feature of these specialized intersection representations is that they can often be used to obtain efficient algorithms for standard combinatorial optimization problems, e.g., it is well-known [5] that the Clique and Independent Set problems, as well as various coloring and Hamiltonicity problems are all efficiently solvable on interval graphs, and the algorithms often leverage on the intersection representation. This led Biró et al. [2] to introduce an elegant family of intersection graph classes, called *H-graphs*, over universes that may be seen as (multi) graphs. Formally, the parameter H is a multigraph, and a graph G is an *H-graph* when there is a *subdivision* H_{sub} of H[1] and a collection $\mathcal{M} = \{M_v \subseteq V(H_{\text{sub}}) \mid v \in V(G)\}$ of sets, where we refer to M_v as the *model of v*, such that

- for every $v \in V(G)$, its model M_v induces a connected subgraph of H_{sub}, and
- $\{u, v\} \in E(G)$ if and only if $M_u \cap M_v \neq \emptyset$.

In this context, H_{sub} *represents* G. Observe that, for any interval graph G, there is a path P (i.e., a subdivision of K_2) such that P represents G meaning that the interval graphs are precisely the K_2-graphs. Similarly, the circular-arc graphs are C-graphs for any cycle C, and every chordal graph is a T-graph for some tree T, i.e., indeed, H-graphs can be seen as a parameterized generalization of several important families of intersection graphs, where H is a parameter reflecting the distance of a graph to an interval graph. Biró et al. [2] provided polynomial-time algorithms (via treewidth-based techniques) for coloring problems on H-graphs for fixed H, but left many interesting problems open.

The classes of H-graphs have seen renewed interest in recent years concerning their structure and recognition [8], relation to other graph parameters [8, 12], and primarily regarding the computational complexity of standard algorithmic problems when parameterized by the size of H [1, 7, 8, 9, 12, 17, 18]. Of particular relevance to our paper is the work on Hamiltonicity problems [7] as it introduces proper H-graphs, which are to *proper interval graphs* as H-graphs are to interval graphs. Namely, for a graph G, a subdivision H_{sub} of H *properly represents* G when H_{sub} represents G using models $\{M_v \subseteq V(H_{\text{sub}}) \mid v \in V(G)\}$ such that for each $u, v \in V(G)$, neither $M_u \subseteq M_v$ nor $M_v \subseteq M_u$. In particular, on proper H-graphs polynomial size kernels (in the size of H) were developed for various Hamiltonicity problems [7], but the recognition problems were left open.

The cornerstone problem for every graph class is recognizability, and we focus on the recognition problem for proper H-graphs both when H is part of the input and when H is fixed. It is important to note that the problem of testing whether for a given graph G and given tree T, the graph G is a T-graph is NP-complete [20, Theorem 4]. In fact, the reduction [20, Theorem 4] also implies that testing whether G is a *proper* T-graph

[1] H_{sub} is obtained from H by iteratively replacing an edge $\{u, v\}$ by a path uwv, where w is a new vertex.

is NP-complete. In contrast to this, it is known that when T is fixed, testing whether a given graph is a T-graph can be done in polynomial-time [8], i.e., XP in the size of T; but it is not known whether the problem is FPT in the size of T. When going beyond trees the recognition problem becomes much harder. Namely, for each fixed non-cactus graph H, H-graph recognition is NP-complete [8]. However, for fixed H, it seems that only two cases of proper H-graph recognition have been studied: The proper interval graphs (proper K_2-graphs) [10, 11] and the proper circular-arc graphs (proper C-graphs, for any cycle C) [11] can each be recognized in linear-time.

Our Contribution. In our main result, we show that the recognition of proper T-graphs is fixed-parameter tractable (FPT) with respect to the size of T by proving the following.

▶ **Theorem 1.** *There is an algorithm that, given an n-vertex graph G and a tree T with t nodes, decides whether G is a proper T-graph, and if yes, outputs a proper T-representation, in $2^{\mathcal{O}(t^2 \log t)} \cdot n^3$ time.*

To obtain our FPT algorithm for proper T-graph recognition, we first observe that the problem can be reduced to the case when the input graph G is connected and chordal. We proceed in the following three key steps.

In Section 3, we introduce compact representations which are an analog to the clique-trees of chordal graphs that incorporates the properness condition. We characterize the proper T-graphs via these compact representations. This allows us to work with maximal cliques of the input graph that can be listed in linear-time due to the chordalilty of G.

In Subsections 4.1 and 4.2, independent of the tree T, we partition the maximal cliques into a collection of the so-called *chains* each one necessarily forming a path in any proper T-representation, and the remaining singleton cliques that are marked and treated separately. We show that having a compact T-representation means there are, in terms of the size of T, at most quadratically many of these marked cliques and chains altogether.

In Subsections 4.3 and 4.4, we combine these ideas to form our FPT algorithm for proper T-graph recognition. First, our algorithm guesses a layout of the chains and the marked maximal cliques. The remaining non-trivial task is to decide whether there is a compact representation corresponding to the guessed layout. We select a root of the tree and show a combinatorial result that if any compact representation realizes some layout, it can be assumed to have some special properties concerning the usage of the nodes of degree at least three of the tree by the models with respect to the root. We call representations satisfying these properties *normalized*. Our algorithm follows the layout bottom-up and constructs a normalized representation if it exists.

We complement our algorithmic result from Theorem 1 by proving that if H is not constrained to be a tree, the recognition problem for proper H-graphs becomes NP-complete even if H has bounded size. This negative result employs a reduction quite similar to the one used for (non-proper) H-graphs in [8], and as such is discussed and proven in the full version.

▶ **Theorem 2** ($\star$). *There is a 4-vertex, 5-edge multigraph $\mathfrak{D}$ (defined by $V(\mathfrak{D}) = \{a, b, c, d\}$ and $E(\mathfrak{D}) = \{ab, bc, bc, bc, cd\}$) such that proper $\mathfrak{D}$-graph recognition is NP-complete.*

Note that this and further statements proven in the full version are marked with ($\star$).

2 Preliminaries

General Notation. We consider undirected graphs G with vertex set $V(G)$ and edge set $E(G)$. Usually we denote an edge as a set $\{u,v\}$. However, when needed, we also denote an edge an ordered pair (u,v). For any subset W of $V(G)$, we use $N(W)$ to denote the *open neighborhood* of W, i.e., $N(W) := \{u \in V(G) \setminus W \mid \{u,w\} \in E(G), w \in W\}$, and for a single vertex $w \in V(G)$, $N(w) := N(\{w\})$. We denote the set of maximal cliques of a graph as $\mathcal{C}(G)$. The shorthand $[n]$ denotes the set $\{1,\ldots,n\}$ of integers.

A *subdivision* H' of a graph H at an edge $\{u,w\}$ is the graph resulting from replacing edge $\{u,w\}$ with a path u,v,w where v is new vertex. A *contraction* of a graph H at an edge $\{u,w\}$ is the graph resulting from removing edge $\{u,v\}$ and identifying the two vertices u and w. Then H_{sub} is a *re-subdivision* of H if it can be obtained by a series of contractions of H (possibly none) followed by a series of subdivisions. In particular a graph G is a (proper) H-graph if and only if there is a re-subdivision that (properly) represents G.

Let T be a tree. For any pair x,y of nodes of T, we denote by $T[x,y]$ the set of nodes of the unique path from x to y in T. Note that $T[x,y] = T[y,x]$. We similarly define $T(x,y] := T[x,y] \setminus \{x\}$ and $T(x,y) := T[x,y] \setminus \{x,y\}$. A tuple of nodes $(x_1,\ldots,x_s)$ is T-*ordered* if there exists a path in the graph T from x_1 to x_s where the nodes $x_1,\ldots,x_s$ occur in this order, i.e., $T[x_1,x_s]$ is the path $x_1,\ldots,x_s$.

***H*-graphs.** Consider a re-subdivision H_{sub} of a graph H that (properly) represents a graph G using models $\{M_v \subseteq V(H_{\text{sub}}) \mid v \in V(G)\}$. For clarity, we refer to each $x \in V(H_{\text{sub}})$ as a *node* and to each $v \in V(G)$ as a *vertex*. We further refer to each node $x \in V(H_{\text{sub}})$ as:

- a *subdivision node* when it has degree two,
- a *branching node* when it has degree more than two, and
- a *leaf node* if it has degree one.

For a set of nodes $X \subseteq V(H_{\text{sub}})$, let $V_X := \{v \in V(G) \mid M_v \cap X \neq \emptyset\}$. When $X = \{x\}$, we also write V_x to mean $V_{\{x\}}$. For a subset of vertices $\Gamma \subseteq V(G)$, let $M_\Gamma := \bigcup_{v\in\Gamma} M_v$. We say that a set Γ of vertices (or nodes) is *connected* if the graph induced by Γ is connected.

▶ **Observation 3.** *Let H_{sub} (properly) represent a graph G. For any connected subset Γ of $V(G)$, the model M_Γ of Γ is connected in H_{sub}.*

Chordal Graphs and Clique Trees. A graph is *chordal* when it does not contain an induced k-vertex cycle for any $k \geq 4$. The chordal graphs are well known to be characterized as the intersection graphs of subtrees of a tree, i.e., for every chordal graph G, there is a tree T that represents G (G is a T-graph) [6, 14, 25]. In fact, G is chordal if and only if there is a tree T with models $\{M_v \subseteq V(T) \mid v \in V(G)\}$ where, for each node $x \in V(T)$, V_x is a maximal clique of G and for every node $y \in V(T)$ with $y \neq x$, $V_y \neq V_x$ [6, 14, 25]. These special representations of G are called *clique trees*, and one can be constructed in linear-time [3, 13]. Note that chordal graphs have a simpler linear-time recognition algorithm [23]. Finally, every chordal graph G has at most n maximal cliques where $n = |V(G)|$ and the sum of the sizes of the maximal cliques of G is $O(n+m)$ [15]. In particular, the total size of a clique tree of G is $O(n+m)$. Clearly, the latter two properties of chordal graphs also apply to (proper) T-graphs independently of T, and we will use them implicitly throughout our discussions.

Each chordal graph G is also a proper T-graph for a tree T. Namely, if a tree T represents G via models $\{M_v \mid v \in V(G)\}$, any tree T' built from T as follows properly represents G: Extend each model M_v by a new node x_v and add $\{x, x_v\}$ to $E(T)$ for some $x \in M_v$.

3 Compact Representations of Proper T-Graphs

In this section we introduce an analogue of clique trees for proper T-graphs. Ideally, G being a proper T-graph would imply a clique tree with the topology of T representing G which satisfies properness; in other words: a re-subdivision T_{sub} of T with models satisfying properness (i.e., forbidding $M_u \subseteq M_v$ for every pair $u, v \in V(G)$) such that every node x represents a unique maximal clique V_x. However, a proper tree-representation of a graph G may use a lot of nodes just to ensure that the models M_u and M_v obey properness; which is already the case for K_2 and its interval representation. Fortunately we may guarantee that almost all nodes represent a unique maximal clique by relaxing the properness condition. Instead of forbidding containment, we require that when M_u intersects M_v, there is a *place* where M_u may be extended (as needed) to break containment. That place is an edge $\{x, y\}$ in the tree T_{sub} where u *strongly escapes* v, that is, $u, v \in V_x$ and $v \notin V_y$. Actually, a weaker version of *escape* suffices. A vertex u *escapes* v if $u \in V_x$ and $v \notin V_y$.

▶ **Definition 4.** *Let a tree T_{sub} with models $\{M_u \mid u \in V(G)\}$ represent a connected graph G. We say that T_{sub} is a* compact representation *of G if*
(C1) *for every leaf node $x \in V(T_{\text{sub}})$, $V_x = \emptyset$,*
(C2) *there is a bijection between the non leaves of $V(T_{\text{sub}})$ and the maximal cliques $\mathcal{C}(G)$, and*
(C3) *for every ordered pair (u, v) with $u, v \in V(G)$, there is an edge $\{x, y\} \in E(T_{\text{sub}})$ where u escapes v.*

▶ **Observation 5** ($\star$). *Let a tree T_{sub} with models $\{M_u \mid u \in V(G)\}$ represent a connected graph G and satisfy condition (C1). For any vertices u, v of G, u and v satisfy the condition (C3) if and only if u and v satisfy condition*
(C3') *if $M_u \cap M_v \neq \emptyset$, then u strongly escapes v.*

Note that, the non-leaves of a compact representation are in one-to-one correspondence with the maximal cliques $\mathcal{C}(G)$. Namely, we identify the non-leaves with the maximal cliques, which implicitly defines the models. Thus, we often omit the explicit statement of the models.

▶ **Observation 6.** *Let G be a connected graph. For any compact representation T_{sub} of G,*
1. *for every distinct non-leaves $x, y \in V(T_{\text{sub}})$ there is a vertex $u \in V_x \setminus V_y$, and*
2. *for every edge $\{x, y\} \in E(T_{\text{sub}})$ of non-leaves x, y, there is a vertex $u \in V_x \cap V_y$.*

We (constructively) show that properness and compactness are essentially equivalent. To obtain compactness from properness, we carefully contract edges where a node was used solely to assure properness. This can involve contracting edges of T when the vertex sets of the nodes of an edge are comparable, e.g., if they are the same maximal clique. To obtain properness from compactness, we subdivide the tree and appropriately extend the models.

▶ **Theorem 7** ($\star$). *For any connected graph G and tree $T \neq K_1$, the graph G is a proper T-graph if and only if there is re-subdivision T_{sub} of T that is a compact representation of G.*

Thus, instead of finding a proper representation, we search for a compact representation. The actual "properness" is hidden in the condition (C3), and we may refer to this condition as *properness*. See also examples in Figure 1.

Our algorithm further relies on the following property of the models M_Γ of the (connected) components of $G - V_y$ for some non-leaf y; see also Figure 2(a). Let $\mathbf{\Gamma}(y)$ (w.r.t. graph G) be the vertex sets of the components of $G - V_y$. We note that $N(\Gamma) \subseteq V_y$ for every $\Gamma \in \mathbf{\Gamma}(y)$. Let a node y be an *eye* if it is a neighbor of a leaf or if it is a branching node.

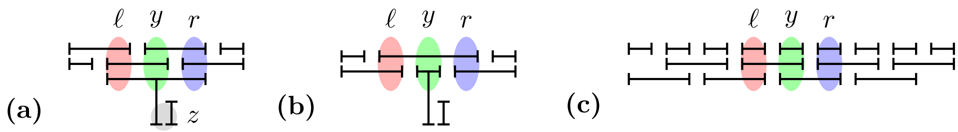

Figure 1 (a) A proper $K_{1,3}$-graph. Triple (ℓ, y, r) is surrounding. Any representation positions y between ℓ and r. Component $V_z \setminus V_\ell$ complies with condition (2B). Further, edge $\{z, y\}$ may be replaced by edge $\{z, \ell\}$ or $\{z, r\}$. (b) A proper $K_{1,3}$-graph. Triple (ℓ, y, r) is not surrounding. A "private" vertex in $V_y \setminus N(\Gamma_\ell) \cup N(\Gamma_r)$ contradicts condition (2A). Indeed, any of ℓ, y, r may realize the branching node. (c) Triple (ℓ, y, r) is surrounding. For $\{\Gamma_\ell, \Gamma_r\} = \mathbf{\Gamma}(y)$ condition (1) allows private vertices in V_y; otherwise, this proper interval graph would have no surrounded nodes.

▶ **Lemma 8** ($\star$). *Let G be a connected graph. For any compact representation T_{sub} of G and any non-leaf node $y \in V(T_{\text{sub}})$,*

1. *$\{y\}$ and M_Γ for $\Gamma \in \mathbf{\Gamma}(y)$ partition the non-leaves of T_{sub}, and*
2. *each partition M_Γ contains an eye, hence $|\mathbf{\Gamma}(y)| \leq |V(T)|$.*

4 Finding a Compact Representation

In this section, we prove Theorem 1; namely, we establish our FPT algorithm. Throughout the discussion, we assume G is connected, and handle disconnected graphs within the final proof. From Section 3, it suffices to check for a compact representation T_{sub}. In Subsection 4.1, we establish the concept of *surrounded* nodes, which leads, in Subsection 4.2, to the *chains* that necessarily form paths in any compact tree representation. We establish that the chains (composed of surrounded nodes), and the remaining non-surrounded nodes are only quadratically many in the size of the desired tree T. In Subsection 4.3, we formalize the way these pieces fit together as *templates*. Finally, Subsection 4.4 contains the algorithm establishing Theorem 1. It proceeds by enumerating candidate templates and (non-trivially) testing whether a template admits a compact representation via a bottom-up procedure.

4.1 Surrounded Nodes

We establish conditions for arbitrary nodes ℓ, y, r that determines the relative position of ℓ, y, r in any representation T_{sub}, a relation which we denote as (ℓ, y, r) surrounding. Clearly, this positioning is unlikely to be possible for every triple (ℓ, y, r) since this would yield a polynomial-time algorithm. However, by carefully crafting our first two requirements, we may still relatively position almost all nodes ℓ, y, r. We only fail for a few nodes y, at most quadratic in the size of the host tree T, hence our parameter.

▶ **Definition 9.** *Consider non-leaves ℓ, y, r of T_{sub}. There is a component $\Gamma_\ell \in \mathbf{\Gamma}(y)$ containing $V_\ell \setminus V_y$, likewise a component $\Gamma_r \in \mathbf{\Gamma}(y)$ containing $V_r \setminus V_y$. Then (ℓ, y, r) is a* surrounding triple, *if the following conditions are met:*

(1) *If $\{\Gamma_\ell, \Gamma_r\} = \mathbf{\Gamma}(y)$, then $V_y = N(\Gamma_\ell) \cup N(\Gamma_r)$ or $N(\Gamma_\ell) \cap N(\Gamma_r) = \emptyset$;*

(2) *if $\{\Gamma_\ell, \Gamma_r\} \subsetneq \mathbf{\Gamma}(y)$,*

  (2A) *$V_y = N(\Gamma_\ell) \cup N(\Gamma_r)$, and*

  (2B) *for every $\Gamma \in \mathbf{\Gamma}(y) \setminus \{\Gamma_\ell, \Gamma_r\}$ we have: $N(\Gamma) \subseteq N(\Gamma_\ell) \cap N(\Gamma_r)$; and*

(3) *for every ℓ', r' that satisfy (1), (2A), and (2B) where $\Gamma_\ell = \Gamma_{\ell'}$ and $\Gamma_r = \Gamma_{r'}$, we have $V_{\ell'} \cap V_y \subseteq V_\ell \cap V_y$ and $V_{r'} \cap V_y \subseteq V_r \cap V_y$.*

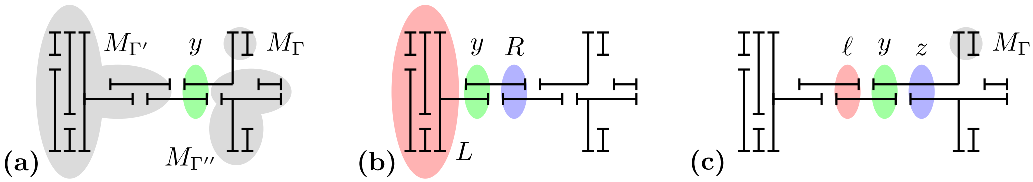

Figure 2 (a) Graph $G - V_y$ splits into three connected components. Their models partition the non-leaves of $T_{\text{sub}} - \{y\}$. Each model contains an eye (of this subdivision of a double-star). (b) Sets L, R exactly capture the nodes ℓ, r that (together with y) form a surrounding triple (ℓ, y, r). Here, y neighbors a subdivision node r to its right. In such a case a y-guard $R \supseteq \{r\}$ may only contain r. (c) Subdivision node y is not surrounded. A remote component Γ falsifies condition (2B). Note that $\mathbf{\Gamma}^{-1}(\Gamma) = \{y, z\}$. Thus Γ does not falsify (2B) for another subdivision node y'.

Note, that the definition does *not* depend on the considered representation T_{sub}. Importantly, $\Gamma_\ell \neq \Gamma_r$ is (implicitly) required by condition (1). We say that y *is surrounded*, if a triple (ℓ, y, r) is a surrounding triple for some nodes ℓ, r. See Figure 1 for examples.

For each node y, the connected components Γ_ℓ and Γ_r satisfy or falsify the first two conditions independently of the precise maximal cliques V_ℓ and V_r. However, condition (3) requires V_ℓ and V_r to be the closest ones to V_y. In many cases condition (3) implies that ℓ and r directly neighbor y. In fact, for a surrounded node y, there are sets of nodes L and R that exactly localize the nodes ℓ and r forming a surrounding triple with y. Formally, L, R are *y-guards*: A set of non-leaves $L \subseteq V(T_{\text{sub}})$ is a *y-guard* if $L \cup \{y\}$ is connected, and y is adjacent to a node $\ell \in L$ such that $\{\ell\} = L$ or ℓ is a branching node of T_{sub}; see Figure 2(b).

▶ **Lemma 10** ($\star$). *Let a tree T_{sub} be a compact representation of a connected graph G. Let y be surrounded. There are distinct y-guards L and R such that (ℓ, y, r) is surrounding if and only if $(\ell, r) \in (L \times R) \cup (R \times L)$. Moreover there is an $\mathcal{O}(t^3 n^3)$-time algorithm that determines sets L, R for every surrounded node y; where $t = |V(T)|$ and $n = |V(G)|$.*

The *guards of y* are such distinct y-guards L and R that precisely characterize its surrounding triples. It is worth noting that a node y that is surrounded by subdivision nodes has singleton guards $\{\ell\}$ and $\{r\}$, which then must be neighbors y in any representation.

Surprisingly there is also a quadratic bound in $|V(T)|$ on the number of *not* surrounded nodes. To show this, the main difficulty is that the conditions (2A) and (2B) fail for a subdivision node y due to some *remote* component Γ, which is a connected component $\Gamma \in \mathbf{\Gamma}(y) \setminus \{\Gamma_\ell, \Gamma_r\}$. Here, let $y \in T_{\text{sub}}(x, z)$ for some neighbors $x, z \in V(T)$. To cope with these remote components, we use three ingredients:

- A component $\Gamma \in \mathbf{\Gamma}(y)$ that falsifies the conditions in question relates to $\mathbf{\Gamma}(x)$ or $\mathbf{\Gamma}(z)$: Its model M_Γ must be outside of $T_{\text{sub}}[x, z]$. By examining the nodes y' that have Γ as a component, it follows that either $\Gamma \in \mathbf{\Gamma}(x)$ or $\Gamma \in \mathbf{\Gamma}(z)$.
- We use Lemma 8 on components $\Gamma \in \mathbf{\Gamma}(x)$, likewise for $\mathbf{\Gamma}(z)$: The models of the components $\Gamma \in \mathbf{\Gamma}(x)$ partition the non-leaves T_{sub}, and each of its models M_Γ contains an eye. That means that $\mathbf{\Gamma}(x)$ contains at most $|E(T)|$ components.
- A component $\Gamma \in \mathbf{\Gamma}(x)$ can be remote for at most one node y on the path $T_{\text{sub}}(x, z)$, and hence falsify the surround conditions for at most one y on that path. This yields a simple $2|E(T_{\text{sub}})|$-bound for not surrounded nodes on that path; see also Figure 2(c).

By incorporating these ideas in a more careful manner we obtain the following bound:

▶ **Lemma 11** ($\star$). *Let subdivision T_{sub} of a tree T be a compact representation of a connected graph G. There are at most $|E(T)|^2 + 1$ non-leaves of $V(T_{\text{sub}})$ that are not surrounded.*

4.2 Chains: Paths in any Representation

As observed previously, singleton guards $\{\ell\}$ and $\{r\}$ of a node y must neighbor y. If a path of nodes $y_1, \dots, y_s$ is made of aligned guards, i.e., $\{y_{i-1}\}$ and $\{y_{i+1}\}$ are the guards of y_i, then it is a path in any representation T_{sub}. In this subsection we define such paths as *chains*. A *chain* captures a maximum length path $y_1, \dots, y_s$ with aligned guards. They also include the initial and final guard Y_0 and Y_{s+1}, their *terminals*.

▶ **Definition 12.** *A* chain *is a maximal length $s \geq 1$ sequence of sets of non-leaf nodes*

$$\mathcal{Y} = \langle Y_0, \{y_1\}, \dots, \{y_s\}, Y_{s+1}\rangle$$

where y_i has guards Y_{i-1} and Y_{i+1} for every $i \in [s]$; and where $Y_i \coloneqq \{y_i\}$ for $i \in [s]$.

To avoid lengthy statements, let us implicitly use $Y_i \coloneqq \{y_i\}$ from now on. Let $I(\mathcal{Y}) = \{y_1, \dots, y_s\}$ be the set of inner nodes of a chain $\mathcal{Y}$. Let $\mathcal{H}(G)$ be the set of chains of a (connected) graph G.

By Lemma 10 such a chain implies that $y_1, \dots, y_s$ is a path in any representation T_{sub}. Also there are unique realizations of the terminals $y_0 \in Y_0 \cap N_{T_{\text{sub}}}(y_1)$ and $y_{s+1} \in Y_{s+t} \cap N_{T_{\text{sub}}}(y_s)$ in a given T_{sub}. Let $y_0 y_1 \dots y_s y_{s+1}$ be the *corresponding* path of $\mathcal{Y}$ in the tree T_{sub}.

The terminals define how the chains may attach to each other. In the very simple case a terminal Y_0 may only consist of a single non-surrounded node, and hence any other chain must attach to that node. Otherwise, as we show later, only the following option remains: Terminal Y_0 contains some surrounded node y'_i which is part of another chain $\langle Y'_0, \dots, Y'_{s'+1}\rangle$. Interestingly Y_0 then contains the whole path $y'_1, \dots, y'_s$. This allows us to freely change the attachment of the chain $\langle Y_0, \{y_1\}, \dots\rangle$ to the chain $\langle \dots, \{y'_i\}, \dots\rangle$ without breaking the connectivity of the representation, though possibly the properness. Similarly, the intersection $V_x \cap V_{y_i}$ for an outside-of-path node x is equal for every $i \in [s]$, and therefore may be reattached in the same sense.

▶ **Lemma 13** (⋆). *Consider a chain $\langle Y_0, \dots, \{y_i\}, \dots, Y_{s+1}\rangle$. A neighbor $x \in N(y_i) \setminus (Y_{i-1} \cup Y_{i+1})$ has $V_x \cap V_{y_i} = V_x \cap V_{y_j}$, for every $j \in [s]$. Furthermore, $Y' \cap I(\mathcal{Y}) \in \{\emptyset, I(\mathcal{Y})\}$ for every terminal Y' of any chain.*

In the following subsection we aim for a bound on the number of chains. We note here that chains behave in a reasonable way: Each surrounded node y is part of exactly one chain, because otherwise it contradicts the classification by y-guards as seen in Lemma 10. Clearly, a chain does not contain a node more than once, since y_i-guards Y_{i-1}, Y_{i+1} are in different subtrees of y_i, for every $i \in [s]$. As the next step, we observe that terminals only consist of either a not-surrounded node or a non-singleton guard, i.e., are from

- $\overline{\mathcal{S}}(G) \coloneqq \{\{y_0\} \mid y_0 \in \mathcal{C}(G) \text{ is not surrounded}\}$, or
- $\overline{\mathcal{U}}(G) \coloneqq \{Y_0 \mid Y_0 \text{ is guard of some } y_1 \in \mathcal{C}(G), |Y_0| > 1\}$.

▶ **Lemma 14** (⋆). *Let T_{sub} be a compact representation of a connected graph G. Then every terminal Y_0, Y_{s+1} of a chain of G is part of $\overline{\mathcal{S}}(G)$ or $\overline{\mathcal{U}}(G)$.*

Further, let the family of inner nodes be $\mathcal{I}(G) \coloneqq \{I(\mathcal{Y}) \mid \mathcal{Y} \in \mathcal{H}(G)\}$. Note here that $\mathcal{I}(G) \cup \overline{\mathcal{S}}(G)$ partition the maximal cliques $\mathcal{C}(G)$.

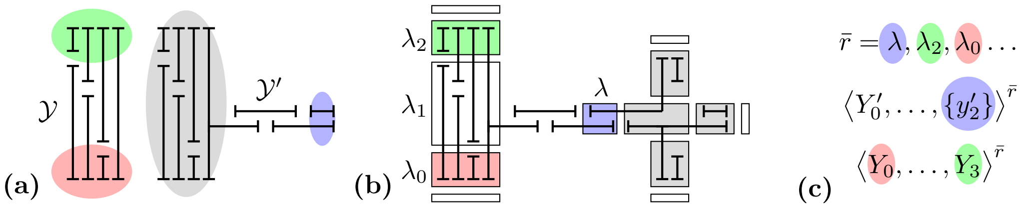

Figure 3 a) Depiction of the chains of almost the graph from Figure 2a) with marked terminals. b) A template prescribes the positioning of terminals, chains and $\overline{\mathcal{S}}$. Empty boxes represent leaves. The chain $\mathcal{Y}$ is mapped to path $\lambda_0\lambda_1\lambda_2$. The chain $\mathcal{Y}'$ is mapped to the single edge path $\lambda_1\lambda$ (an edge despite the gap in the picture). We have $t^0(\lambda_1) = I(\mathcal{Y})$, and indeed any attachment of $\mathcal{Y}'$ to an inner node of $\mathcal{Y}$ is possible. For the chain $\mathcal{Y}'$ a mapping to single edge suffices since here it is realized by a path of subdivision nodes. c) A sample root-ordering (colors mark the mapping of t^0), and resulting orientation of chains $\mathcal{Y}'$ and $\mathcal{Y}$. Here λ_2 (mapped to Y_3) is a tie-breaker for $\mathcal{Y}$.

4.3 Template: Fixing the Topology of Chains

The set of chains $\mathcal{H}(G)$ of a (connected) graph G already considerably prescribes many paths that are present in any proper representation T_{sub} of G. What remains are two problems of a more global flavor: For a chain there may be a vast range of possible connections. Simultaneously we have to assure properness, i.e., that any vertices u and v escape each other. To cope with these tasks we define a preliminary representation, a *template*. A template considerably fixes the topology of a tree T_{sub} representing G. It narrows down the possible representations such that we can focus on the properness. At the same time, our final algorithm has to guess a template, thus its possibilities should be bounded by our parameter, the size of T.

To fix the relative positions of chains, a template locates the terminals of a chain, Y_0 and Y_{s+1}, on some template tree T^0. A concrete realization T_{sub} of that template is a subdivision of T^0. It realizes a chain between its terminals as prescribed by the template. More precisely, t^0 maps the nodes λ of T^0 to the terminals of chains. To avoid ambiguity, let $t^0(\lambda)$ not map to a mere terminal Y_0, if $Y_0 \in \overline{\mathcal{U}}(G)$ (as it may be huge), but narrow down the mapping to some set of inner nodes of $\mathcal{I}(G)$. In other words we fix the neighborhood of a chain on the "chain-level". Note that any $Y_0 \in \overline{\mathcal{U}}(G)$ is a superset of some set of inner nodes, as seen in Lemma 13. For convenience, let us also fix a mapping h^0 of chains $\langle Y_0, \{y_1\}, \ldots, \{y_s\}, Y_{s+1}\rangle$ onto T^0: Let h^0 map to paths $\lambda_0, \ldots, \lambda_{s'+1}$ in T^0, which should be conforming with the terminals, which is $t^0(\lambda_0) \subseteq Y_0$ and $t^0(\lambda_{s'+1}) \subseteq Y_{s+1}$. If the chain does not contain any branching node, it suffices to represent the terminals. Then h^0 maps simply to the single-edge path $\lambda_0, \lambda_{s'+1}$ (for example $\mathcal{Y}'$ in Figure 3(a),(b)). In the other extreme, every inner node may be a branching node (respectively used as a terminal of another chain), thus possibly $s' = s$.

In more detail, a chain $\mathcal{Y}$ may correspond to a path $y_0 \ldots y_{s+1}$ with an inner branching node y_0' which is the endpoint of a path $y_0'y_1' \ldots$ corresponding to another chain $\mathcal{Y}' = \langle Y_0', \ldots, \rangle$. Thus, the chain $\mathcal{Y}$ must be mapped to a path with an inner node λ_j that is an endpoint of the path $\lambda_j, \lambda_1' \ldots, \lambda_{s''+1}'$ that is the image of the other chain $\mathcal{Y}'$ (for example $\mathcal{Y}$ in Figure 3a)b)). As seen before, then $I(\mathcal{Y}) \subseteq Y_0' \in \overline{\mathcal{U}}(G)$. Hence, the mapping of that terminal is $t^0(\lambda_j) = I(\mathcal{Y})$. We require this behavior for inner nodes like λ_j.

▶ **Definition 15.** *Let G be a connected chordal graph. A* template *of a tree T (w.r.t. G) is a triple (T^0, t^0, h^0) where*

- *T^0 is a re-subdivision of T,*
- *t^0 is a mapping of the non-leaves of T^0 to $\overline{\mathcal{S}}(G) \cup \mathcal{I}(G)$,*
- *h^0 is a bijection of the chains $\mathcal{H}(G)$ to an edge-disjoint set of non-trivial (i.e., containing at least one edge) paths between non-leaves of T^0, and*
- *for every chain $\langle Y_0, \ldots, Y_{s+1}\rangle \in \mathcal{H}(G)$ mapped to a path $\lambda_0, \ldots, \lambda_{s'+1}$ we have that $t^0(\lambda_0) \subseteq Y_0$ and $t^0(\lambda_{s'+1}) \subseteq Y_{s+1}$ and $t^0(\lambda_i) = I(\mathcal{Y})$ for every $i \in [s']$.*

Consider a tree T_{sub} where each non-leaf y is identified with a maximal clique V_y. Then T_{sub} realizes *the template (T^0, t^0, h^0) if T_{sub} results from subdividing T^0 and $V_\lambda \in t^0(\lambda)$ for every non-leaf λ of T^0.*

Notably the image of h^0 does not necessarily cover every edge between non-leaves. Namely, non-surrounded nodes y and y' might be neighbors. The next lemma establishes that every tree T_{sub} that is a compact representation realizes some template, as we intended.

▶ **Lemma 16 (⋆).** *If T_{sub} is a re-subdivision of a tree T and a compact representation of a connected graph G, then T_{sub} realizes some template (T^0, t^0, h^0) of T.*

As formalized in the next lemma, we can enumerate the possible templates in FPT, since the number of chains is quadratically bounded in $V(T)$.

▶ **Lemma 17 (⋆).** *There are $2^{\mathcal{O}(t^2 \log t)}$ possible templates of a tree T w.r.t. a connected chordal graph G, which can be enumerated in time $2^{\mathcal{O}(t^2 \log t)} \cdot n^3$; where $t = |V(T)|$, $n = |V(G)|$.*

4.4 Normalized Representation: Achieving Properness

We now consider a fixed template and focus on the properness. The remaining leeway is to locally change the branching nodes of a particular chain. We use a construction which only fails if the considered template does not allow a compact representation. The result is a *normalized* representation.

Any representation T_{sub} can be normalized by a bottom-up process: Move each branching node y_i up as much as possible within the local subtree, i.e. as long as the subtree remains compact. By moving up, we mean replacing y_i by y_j as a branching node that is closer to a global root in the chain. The set of nodes that potentially replace y_i behave in a linear fashion, and hence allow this greedy approach. Thus we may assume that a normalized representation exists for a yes-instance. Our algorithm though has to construct a representation from scratch. By incorporating this idea in a more careful manner we may assemble each subtree of a normalized T_{sub} bottom-up. Here we attach the inductive subtrees in the most conservative way; then the same normalization step as before yields the desired new subtree. Again, the linear behavior of the potential replacements enable this greedy approach.

To start, let us define the root of a template. Since chains may not "align" towards a picked root $\bar{r}_1$, we have to work with additional tie-breakers $\bar{r}_2, \bar{r}_3, \ldots$.

▶ **Definition 18.** *A* root-ordering *$\bar{r}$ is an ordering $\bar{r}_1, \bar{r}_2, \ldots$ of nodes $V(T^0) \cap \{\lambda \mid t^0(\lambda) \in \overline{\mathcal{S}}\}$.*

The specific root-ordering will not be of importance and we may pick one arbitrarily. We assume in the following that every tree and template comes with a root-ordering. See Figure 3 for an example.

▶ **Definition 19.** *A root-ordering $\bar{r}$ and a template (T^0, t^0, h^0) define an orientation for every chain $\mathcal{Y} = \langle Y_0, \ldots, Y_{s+1}\rangle$ as follows. Let $h^0(\mathcal{Y})$ map to a path in T^0 with end nodes λ_0 and λ_{s+1} where $t^0(\lambda_0) \subseteq Y_0$ and $t^0(\lambda_{s+1}) \subseteq Y_{s+1}$. Let k be the smallest index such that $(\lambda_0, \lambda_{s+1}, \bar{r}_k)$ or $(\bar{r}_k, \lambda_0, \lambda_{s+1})$ is T^0-ordered. If $(\lambda_0, \lambda_{s+1}, \bar{r}_k)$ is T^0-ordered, then $\mathcal{Y}$* is oriented towards *Y_{s+1}, which we denote by writing $\langle Y_0, \ldots, Y_{s+1}\rangle^{\bar{r}}$.*

Note that the index k always exists, since every neighbor of a leaf is not surrounded.

Let $R[y_i, y_j]T_{\text{sub}}$ be the tree resulting from replacing branching node y_i by y_j. Its local version is $\rho[y_i, y_j]T_{\text{sub}}$. The models living in the more restrict subtree $\rho^{\downarrow}[y_i, y_j]T_{\text{sub}}$ are critical: Their properness is at stake. We define the possible replacements of a node y_i resulting in a proper representation as the potential $\Phi(T_{\text{sub}}, y_i)$.

▶ **Definition 20.** *Let $\bar{r}$ be a root-ordering. Consider a branching node $y_i \in V(T_{\text{sub}})$ and its chain $\langle Y_0, \ldots, \{y_i\}, \ldots, \{y_j\}, \ldots, Y_{s+1}\rangle^{\bar{r}}$ where y_0 realizes Y_0. For integers $i \leq j < s$, let*

- *$R[y_i, y_j]T_{\text{sub}}$ be the tree T_{sub} where y_i replaces y_j as a branching node, i.e., edge $\{y_i, z\}$ is replaced by a new edge $\{y_j, z\}$, for every node $z \in N_{T_{\text{sub}}}(y_i) \setminus (Y_{i-1} \cup Y_{y+1})$;*
- *$\rho[y_i, y_j]T_{\text{sub}}$ be the tree consisting of the subtree of $R[y_i, y_j]T_{\text{sub}}$ rooted at y_0 (w.r.t. global root $\bar{r}_1$) and path $y_0, \ldots, y_s$ where for every chain node $y_{i'} \in \{y_1, \ldots, y_s\}$ and non-chain neighbor $z' \in N_{R[y_i, y_j]T_{\text{sub}}} \setminus (Y_{i'-1} \cap Y_{i'+1})$ a new leaf node $y'_{i'}$ added adjacent to $y_{i'}$;*
- *$\rho^{\downarrow}[y_i, y_j]T_{\text{sub}}$ be the tree consisting of the subtree of $R[y_i, y_j]T_{\text{sub}}$ rooted at y_0 (w.r.t. global root $\bar{r}_1$) and path $y_0, \ldots, y_{j-1}$.*

For convenience, let $\rho^{\downarrow}[y_i]T_{\text{sub}} \coloneqq \rho^{\downarrow}[y_i, y_i]T_{\text{sub}}$ as well as $\rho[y_i]T_{\text{sub}} \coloneqq \rho[y_i, y_i]T_{\text{sub}}$.

▶ **Definition 21.** *We define the* potential $\Phi(T_{\text{sub}}, y_i)$ *(w.r.t. a template (T^0, t^0, h^0) and root-ordering $\bar{r}$) of non-leaf node y_i.*

- *For a not-surrounded node y_i, let $\Phi(T_{\text{sub}}, y_i) = \{y_i\}$.*
- *For a surrounded branching node y_i, consider its chain $\langle \cdot, \ldots, \{y_i\}, \ldots, \{y_s\}, \cdot\rangle^{\bar{r}}$. The potential $\Phi(T_{\text{sub}}, y_i)$ contains every node $y_j \in \{y_i, \ldots, y_s\}$ where the tree $R[y_i, y_j]T_{\text{sub}}$ is such that every vertex u with model $M_u \subseteq V(\rho^{\downarrow}[y_i, y_j]T_{\text{sub}})$ escapes every other vertex v.*

A simple example is that $y_i \in \Phi(T_{\text{sub}}, y_i)$ for compact representations T_{sub}, as the considered replacement does nothing. In contrast, $\Phi(T_{\text{sub}}, y_i) = \emptyset$ indicates non-properness for the subtree of y_{i-1}. Indeed, the potential of y_i captures exactly the possible replacements of y_i as a branching node.

If some replacement y_j of y_i already is a branching node, the topology changes and $R[y_i, y_j]T_{\text{sub}}$ does not realize the same template. To avoid such issues, we require (without loss of generality) a minimal representation: A tree T_{sub} is *minimal* if there is no compact representation T'_{sub} of G that is a re-subdivision of T_{sub} with fewer branching nodes. Clearly, if there is a representation T_{sub} of G, we may also assume that it is minimal. In particular, the contraction would result in different candidate re-subdivision of T, which we consider separately.

We may compute it locally, meaning it suffices to consider the subtree $\rho[y_i]$. Since the potential $\Phi(T_{\text{sub}}, y_i)$ is a connected subsequence of $\langle y_i, \ldots, y_s\rangle$, we either view it as a set or as such a subsequence. Further, if the potential is $\langle y_i, \ldots, y_j\rangle$, then replacing y_i with the last node y_j makes the resulting potential at y_j singleton. Finally, the potential is independent from later replacements, assuming a bottom-up (i.e., leaf-to-root) procedure.

▶ **Lemma 22** (⋆)**.** *Let T_{sub} be a minimal compact representation of a connected graph G. We observe the following for a chain $\langle \cdot, \ldots, \{y_i\}, \ldots, \{y_j\}, \ldots, \{y_s\}, \cdot\rangle^{\bar{r}}$ for $i \leq j \leq s$:*

1. *If $y_j \in \Phi(T_{\text{sub}}, y_j)$, then $R[y_i, y_j]T_{\text{sub}}$ is a minimal compact representation of G.*
2. *locality, $\Phi(T_{\text{sub}}, y_i) = \Phi(\rho[y_i]T_{\text{sub}}, y_i)$,*
3. *connectivity, $\Phi(T_{\text{sub}}, y_i)$ is connected in T_{sub}, and hence some subsequence $\langle y_i, \ldots, y_j\rangle$,*
4. *linearity, $\Phi(T_{\text{sub}}, y_i) = \langle y_i, \ldots, y_j\rangle$ if and only if $\Phi(R[y_i, y_j]T_{\text{sub}}, y_j) = \langle y_j\rangle$.*
5. *independence, $\Phi(R[y_i, y_j]T_{\text{sub}}, x) \subseteq \Phi(T_{\text{sub}}, x)$ for every node $x \in V(T_{\text{sub}})$ where $(x, y_i, \bar{r}_1)$ is T_{sub}-ordered.*

Consider a tree T_{sub} that realizes a template (T^0, t^0, h^0), and has some root-ordering $\bar{r}$. We say T_{sub} is *normalized* for a node y (w.r.t. to (T^0, t^0, h^0) and $\bar{r}$) if $\Phi(T_{\mathrm{sub}}, y) = \langle y \rangle$. By the locality property, this is equivalent to $\Phi(\rho[y]T_{\mathrm{sub}}, y) = \langle y \rangle$, hence it suffices to consider the local subtree. The whole tree T_{sub} is normalized if it is normalized for every branching node. Now the independence of the potential as explored earlier allows normalizing any representation by a bottom-up procedure. Thus, in a yes-instance, we may assume a normalized representation.

▶ **Lemma 23** ($\star$). *There is an $\mathcal{O}(n^3)$ time algorithm that, given a connected chordal n-vertex graph G and a template (T^0, t^0, h^0), decides whether there is a minimal compact representation of G that realizes (T^0, t^0, h^0), and if one exists, it outputs one that is also normalized.*

Proof (Sketch). Assuming a yes-instance, there is minimal compact representation T'_{sub} of G that realizes template (T^0, t^0, h^0). We may also assume that T'_{sub} is normalized (proven in the full version). Our algorithm outputs a representation isomorphic to T'_{sub}, thus a normalized one as desired. If, however, our construction fails at some point, we correctly conclude that no such representation exists. In the rest of the proof we fix an arbitrary root-ordering $\bar{r}$.

We fix an ordering $\sigma = \lambda_1, \lambda_2, \dots$ of the non-leaf nodes of the template tree T^0, which follows the ordering within in a chain and otherwise is bottom-up. Pick a node λ_k where every non-leaf child of λ_k has been added before, and append it to the ordering. If there is a chain $\langle Y_0, \dots, Y_{s'+1} \rangle^{\bar{r}}$ mapped by h^0 to a path of form $\lambda_0, \lambda_k, \lambda_{k,1}, \dots, \lambda_{k,s'+1}$, append nodes $\lambda_{k,1}, \dots, \lambda_{k,s'+1}$ as well. Then continue to picking a new node until all nodes are ordered.

For $k \geq 1$, let T_k be the subtree of T'_{sub} induced by $\lambda_1, \dots, \lambda_k$, every subdivision node between nodes from $\lambda_1, \dots, \lambda_k$ and leaves neighboring $\lambda_1, \dots, \lambda_k$. By induction over $k \geq 1$, we prove that a tree isomorphic to T_k is polynomial time computable given G, $(T^0, t^0, h^0,)$ and $\bar{r}$. Eventually this yields to a representation T_{sub} isomorphic to T'_{sub}, thus normalized minimal compact and realizing (T^0, t^0, h^0), as desired.

(Induction base, when λ_k neighbors a leaf (w.r.t. to root $\bar{r}_1$)) The node λ_k represents a not-surrounded node $t^0(\lambda_k) = \{\lambda_k\} \in \overline{\mathcal{S}}(G)$. Then T_k consists only of λ_k adjacent to some leaf. Thus, this tree is prescribed by (T^0, t^0, h^0) and hence no computation is required. The induction step where $\lambda_k \in V(T^0)$ is a node with $t^0(\lambda_k) = \{\lambda_k\} \in \overline{\mathcal{S}}(G)$ is similar, and omitted here.

(Induction step $\mathcal{I}(G)$) We consider the case where the template node λ_k is a surrounded branching node. This means that $t^0(\lambda_k) = \{y_1, \dots, y_s\} = I(\mathcal{Y})$ for some chain $\mathcal{Y}$. The template maps $\mathcal{Y}$ to a non-trivial path $\lambda_0, \dots, \lambda_{s'+1}$ in T^0 containing λ_k:

$$\lambda_0 \dots \lambda_{c_0} \lambda_k \lambda_{c'_0} \dots \lambda_{s'+1} \;=\; h^0\big(\langle Y_0, \{y_1\} \dots, \{y_i\}, \dots, \{y_s\}, Y_{s+1} \rangle^{\bar{r}}\big).$$

For each of those inner template nodes $\lambda_{i'}$, we have $t^0(\lambda_{i'}) = \{y_1, \dots, y_s\}$. Let us assume that $t^0(\lambda_0) \subseteq Y_0$ such that the directions of increasing indices match.

Note that λ_{c_0} is ordered before λ_k because of how the ordering σ is defined. Our algorithm may determine λ_{c_0} as the child in T^0 where $(\lambda_0, \lambda_{c_0}, \lambda_k, \lambda_{s'+1})$ is T^0-ordered (possibly $\lambda_0 = \lambda_{c_0}$). The tree T_k realizes λ_k with some inner node y_j with $j \in [s]$. Our task is to determine j without knowing T_k. Let $\lambda_{c_1}, \dots, \lambda_{c_z}$ be the (possibly non-existent, possibly containing $\lambda_{c'_0}$) remaining children of λ_k in T^0. By the induction hypothesis, the subtrees $T_{c_0}, T_{c_1}, \dots, T_{c_z}$ are polynomial time computable.

The tree T_{c_0} realizes λ_{c_0} with some node y_{i-1} for $i \in \{2, \dots, s\}$ where $y_0 \in Y_0$. Because T_{c_0} is a subtree of T_k, this limits the possible realizations of y_j to $\{y_i, \dots, y_s\}$. Let $\overrightarrow{c_0}$ be the path $(y_{i-1}, y_i, \dots, y_s)$.

Consider the adjacency $\lambda_{c_1}\lambda_k$. A simple case is that $t^0(\lambda_{c_1}) = \{\lambda_{c_1}\} \in \overline{\mathcal{S}}(G)$. Then in the tree T_k the two realizing nodes λ_{c_1} and λ_k must be adjacent, and we define $\overrightarrow{c_1}(\lambda_k)$ to be the path $\lambda_{c_1}\lambda_k$. Note that we define the path with a variable λ_k (as named to coincide with the template node since it shares the same variability). For example $\overrightarrow{c_1}(y_j)$ is the path contained in the tree T_k (which we aim to construct).

Otherwise, the node $t^0(\lambda_{c_1})$ is part of a chain with terminal $Y'_{s'+1}$ where $t^0(\lambda_k) \subseteq Y'_{s'+1}$. Now either λ_{c_1} is the other terminal of chain $\mathcal{Y}_1$, or it is the set of inner nodes $I(\mathcal{Y}_1)$ and hence realized as one of them. Thus the format of the chain is either

- $\langle Y'_0, \{y^1_{c_1}\}, \ldots, \{y^{s_1}_{c_1}\}, Y'_{s'+1}\rangle^{\bar{r}}$ where $t^0(\lambda_{c_1}) \subseteq Y'_0$, or
- $\langle \cdot, \ldots, \{y_{c_1}\}, \{y^1_{c_1}\}, \ldots, \{y^{s_1}_{c_1}\}, Y'_{s'+1}\rangle^{\bar{r}}$, where y_{c_1} is the realization of λ_{c_1} in the tree T_{c_1}.

Let $\overrightarrow{c_1}(\lambda_k)$ be the path $(\lambda_{c_1}, y^1_{c_1}, \ldots, y^{s_1}_{c_1}, \lambda_k)$, similarly as before with variable λ_k. For example $\overrightarrow{c_1}(y_j)$ is the path contained in the unknown tree T_k. We define the paths $\overrightarrow{c_2}(\lambda_k), \ldots, \overrightarrow{c_z}(\lambda_k)$ for the other children analogously. Clearly, the same observations apply.

We define the tree $T(\lambda_k)$ similarly. Namely, $T(\lambda_k)$ is the tree containing the subtrees $T_{c_0}, T_{c_1}, \ldots, T_{c_z}$ together with paths $\overrightarrow{c_1}(\lambda_k), \ldots, \overrightarrow{c_z}(\lambda_k)$ and path $y_{i-1}, y_i, \ldots, y_s$. Then T_k is the subtree of $T(y_j)$ rooted at y_j. Thus it remains to determine y_j without knowing T_k.

For that purpose, consider the tree $T(y_i)$, the tree with the most conservative realization of λ_k. Applying the rehang operation yields $R[y_i, y_j]T(y_i) = T(y_j)$. Assume that node y_k of all the nodes of T_k has the smallest distance to the global root $\bar{r}$ (the general case is handled by a slight modification to $T(y_i)$, see full version). Then, since T'_{sub} is normalized and because of locality, we have $\langle y_j\rangle = \Phi(T'_{\text{sub}}, y_j) = \Phi(\rho[y_i]T'_{\text{sub}}, y_j) = \Phi(\rho[y_i]T(y_j), y_j) = \Phi(T(y_j), y_j)$.

Then by the linearity of the potential we have that $\Phi(T(y_i), y_i) = \langle y_i, \ldots, y_j\rangle$. This is how we algorithmically determine y_j, assuming a yes-instance. Thus the desired tree $T_k(y_j)$ is polynomial time computable given graph G, template (T^0, t^0, h^0) and $\bar{r}$. If our algorithm observes that $\Phi(T(y_i), y_i) = \emptyset$ at some point, it contradicts the existence of a normalized representation T'_{sub}, and our algorithm returns no. ◀

Now we outline our FPT algorithm for the parameter $t = |V(T)|$. We assume without loss of generality that G is a chordal graph and $T \neq K_1$ as the problem is trivial otherwise. Note that chordality can be tested in linear time [23]. If G is not connected, each proper interval graph component always be represented using a subdivision of an edge incident to a leaf of T. Thus, these components, which can be recognized in linear time [10, 11], can be excluded from the further consideration. Each of the remaining components is not a proper interval graph and, as such, contains a vertex whose model includes a branching node of T. Thus, if these components number more than the number of branching nodes, G has no T-representation. Assume that this is not the case. We guess an assignment of the connected components of G to connected subtrees of T representing them. Two such subtrees may share an edge (which can be needed to represent both components of G using the end-nodes of this shared edge). Note that are at most $2^{\mathcal{O}(t \log t)}$ possible mappings, and then we can deal with every component of G and the corresponding subtree of T separately. From now on, we assume that G is connected. By Theorem 7, we may look for a compact representation T_{sub}; further, it suffices that T_{sub} is minimal. Therefore, there is a template (T^0, t^0, h^0) that allows a representation of G as seen in Lemma 16. We compute the chains of G and try every template in time $2^{\mathcal{O}(t^2 \log t)} \cdot n^3$ where $n = |V(G)|$, as seen in Lemma 17. Pick an arbitrary root-ordering $\bar{r}$. Then test in polynomial time whether a minimal compact representation of G realizing this template by using Lemma 23. In a positive case, applying Theorem 7 leads to a proper representation. This implies our main result (restated here).

▶ **Theorem 1.** *There is an algorithm that, given an n-vertex graph G and a tree T with t nodes, decides whether G is a proper T-graph, and if yes, outputs a proper T-representation, in $2^{\mathcal{O}(t^2 \log t)} \cdot n^3$ time.*

5 Concluding Remarks and Open Problems

Our recognition algorithm for proper tree-graphs provides the following side result on proper leafage (introduced by Lin et al. [21] analogously to leafage): The *proper leafage* $\ell^\star$ of a chordal graph G is the minimum number of leaf nodes of all trees T that properly represent G. The side result, as in Corollary 24, is that computing the proper leafage is FPT. For the decision version, if G is not a proper interval graph, we simply guess the host tree T_{sub} of minimal leafage and verify properness with our algorithm from Theorem 1. Of course, it still remains open whether computing proper leafage is NP-hard.

▶ **Corollary 24.** *Computing the proper leafage $\ell^\star$ of a chordal graph G is FPT w.r.t. $\ell^\star$.*

While we have shown that proper T-graph recognition is FPT, it remains open whether non-proper T-graph recognition is FPT. Perhaps most importantly, gaps remain concerning the precise conditions under which (proper) H-graph recognition is NP-complete for fixed H.

References

1 Deniz Ağaoğlu and Petr Hliněný. Isomorphism problem for S_d-graphs, 2019. arXiv:1907.01495.

2 Miklós Biró, Mihály Hujter, and Zsolt Tuza. Precoloring extension. I. Interval graphs. *Discrete Mathematics*, 100(1):267–279, 1992.

3 Jean R. S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computation*, pages 1–29, New York, NY, 1993. Springer New York.

4 M. L. Brady and M. Sarrafzadeh. Stretching a knock-knee layout for multilayer wiring. *IEEE Transactions on Computers*, 39(1):148–151, January 1990. doi:10.1109/12.46293.

5 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999. doi:10.1137/1.9780898719796.

6 Peter Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205–212, 1974. doi:10.1016/0012-365X(74)90002-8.

7 Steven Chaplick, Fedor V. Fomin, Petr A. Golovach, Dusan Knop, and Peter Zeman. Kernelization of graph hamiltonicity: Proper H-graphs. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *WADS 2019*, volume 11646 of *LNCS*, pages 296–310. Springer, 2019. doi:10.1007/978-3-030-24766-9_22.

8 Steven Chaplick, Martin Toepfer, Jan Voborník, and Peter Zeman. On H-topological intersection graphs. In *WG 2017*, pages 167–179, 2017. doi:10.1007/978-3-319-68705-6_13.

9 Steven Chaplick and Peter Zeman. Combinatorial problems on H-graphs. In *EUROCOMB'17*, volume 61 of *ENDM*, pages 223–229. Elsevier, 2017. doi:10.1016/j.endm.2017.06.042.

10 Derek G. Corneil. A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Applied Mathematics*, 138(3):371–379, 2004. doi:10.1016/j.dam.2003.07.001.

11 Xiaotie Deng, Pavol Hell, and Jing Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing*, 25(2):390–403, 1996. doi:10.1137/S0097539792269095.

12 Fedor V. Fomin, Petr A. Golovach, and Jean-Florent Raymond. On the tractability of optimization problems on H-graphs. *Algorithmica*, 2020. doi:10.1007/s00453-020-00692-9.

13 Philippe Galinier, Michel Habib, and Christophe Paul. Chordal graphs and their clique graphs. In *WG 1995*, volume 1017 of *LNCS*, pages 358–371. Springer, 1995. doi:10.1007/3-540-60618-1_88.

14 Fănică Gavril. The intersection graphs of subtrees of trees are exactly the chordal graphs. *Journal of Combinatorial Theory Series B*, 16:47–56, 1974.

15 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier Science B.V., Amsterdam, second edition, 2004. With a foreword by Claude Berge.

16 M. L. Huson and A. Sen. Broadcast scheduling algorithms for radio networks. In *Military Communications Conference, 1995. MILCOM '95, Conference Record, IEEE*, volume 2, pages 647–651 vol.2, November 1995. doi:10.1109/MILCOM.1995.483546.

17 Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. Mim-width III. graph powers and generalized distance domination problems. *Theoretical Computer Science*, 796:216–236, 2019. doi:10.1016/j.tcs.2019.09.012.

18 Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-width II. the feedback vertex set problem. *Algorithmica*, 82(1):118–145, 2020. doi:10.1007/s00453-019-00607-3.

19 Deborah Joseph, Joao Meidanis, and Prasoon Tiwari. Determining DNA sequence similarity using maximum independent set algorithms for interval graphs. In Otto Nurmi and Esko Ukkonen, editors, *SWAT '92*, volume 621 of *LNCS*, pages 326–337. Springer, 1992. doi:10.1007/3-540-55706-7_29.

20 Pavel Klavík, Jan Kratochvíl, Yota Otachi, and Toshiki Saitoh. Extending partial representations of subclasses of chordal graphs. *Theoretical Computer Science*, 576:85–101, 2015.

21 In-Jen Lin, Terry A. McKee, and Douglas B. West. The leafage of a chordal graph. *Discuss. Math. Graph Theory*, 18(1):23–48, 1998. doi:10.7151/dmgt.1061.

22 Fred S Roberts. *Graph theory and its applications to problems of society*. SIAM, 1978.

23 Donald J. Rose, Robert Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.

24 F. W. Sinden. Topology of thin film rc circuits. *Bell System Technical Journal*, 45(9):1639–1662, 1966. doi:10.1002/j.1538-7305.1966.tb01713.x.

25 James R. Walter. Representations of chordal graphs as subtrees of a tree. *Journal of Graph Theory*, 2(3):265–267, 1978. doi:10.1002/jgt.3190020311.