Acta Math. Univ. Comenianae
Vol. LXXXVIII, 3 (2019), pp. 651–658

RECOGNISING THE OVERLAP GRAPHS OF SUBTREES OF RESTRICTED TREES IS HARD

J. ENRIGHT AND M. PERGEL

ABSTRACT. The overlap graphs of subtrees in a tree (SOGs) generalise many other graphs classes with set representation characterisations. The complexity of recognising SOGs in open. The complexities of recognising many subclasses of SOGs are known. We consider several subclasses of SOGs by restricting the underlying tree. For a fixed integer $k \geq 3$, we consider:

- The overlap graphs of subtrees in a tree where that tree has k leaves.
- The overlap graphs of subtrees in trees that can be derived from a given input tree by subdivision and have at least three leaves.
- The overlap and intersection graphs of paths in a tree where that tree has maximum degree k.

We show that the recognition problems of these classes are NP-complete. For all other parameters we get circle graphs, well known to be polynomially recognizable.

A graph $G = (V, E)$ with a vertex set $V = \{v_1, \ldots, v_n\}$ and the edge set $E = \{e_1, \ldots, e_m\}$ is an intersection graph of a set system $\{s_1, \ldots, s_n\}$, where for all i, $s_i \subseteq \mathcal{S}$, each vertex v_i corresponds to a set s_i and each edge $e = (v_i, v_j)$ is equivalent to the fact that $s_i \cap s_j \neq \emptyset$. Similarly, a graph $G = (V, E)$ with a vertex set $V = \{v_1, \ldots, v_n\}$ and the edge set $E = \{e_1, \ldots, e_m\}$ is an overlap graph of a set system $\{s_1, \ldots, s_n\}$, where for all i, $s_i \subseteq \mathcal{S}$, each vertex v_i corresponds to a set s_i and each edge $e = (v_i, v_j)$ is equivalent to the fact that $s_i \cap s_j \neq \emptyset$ and neither $s_i \subset s_j$ nor $s_j \subset s_i$.

When we consider the overlap and intersection graphs of particular types of set systems, we define graph classes. Part of the theoretical interest in geometric intersection and overlap graphs stems from efficient algorithms for otherwise NP-hard problems on these graph classes. Often, these algorithms require as input a set intersection representation of a particular type. Thus we are interested in whether or not a given graph has a particular type of intersection representation. This is called the *recognition problem.*

Probably the oldest intersection-defined graphs are *interval graphs*, the *intersection graphs of interval on a line* [**8**]. The interval graphs are generalised by *intersection graphs of paths in a tree* [**4, 12**]. Intersection graphs of paths in a tree are in turn generalised by *chordal graphs*. While primarily defined as the graphs

Received May 24, 2019.
2010 *Mathematics Subject Classification.* Primary 97P20, 97K20, 97K30, 03D15.
This research was partially supported by the Czech Science Foundation grant GA19-08554S.

without induced cycles of length greater than three, chordal graphs are also exactly the *intersection graphs of subtrees in a tree* [**6**]. The overlap analogue of chordal graphs is the class of *subtree overlap graphs*, the *overlap graphs of subtrees in a tree*. Subtree overlap graphs generalise many set representation characterised classes, including chordal graphs and therefore interval graphs.

Gavril [**7**] defined *interval filament graphs* and *subtree filament graphs* as intersection graphs of *filaments on intervals* and *filaments on subtrees*, respectively. *Filaments* are *curves above some geometric structure* (in this case above intervals or subtrees) such that filaments above disjoint structures must not intersect, while filaments above overlapping structures (*i.e.,* over sets a and b such that $a \cap b, a \setminus b$ and also $b \setminus a$ are non-empty) must mutually intersect.

Interval filament graphs are a subclass of subtree overlap graphs, and subtree filament graphs are exactly subtree overlap graphs [**5**]. Given a set representation, we can solve some otherwise hard problems on these classes, including many problems on chordal graphs [**14**], maximum weighted clique and independent set on interval filament graphs and maximum weighted independent set for subtree filament graphs [**7**].

Recognising interval filament graphs is known to be hard [**7, 13**]. In contrast, we can recognise interval graphs and chordal graphs in linear time [**3, 14**], and intersection graphs of paths in a tree in $O(nm)$ time, where n is the number of vertices and m the number of edges in the input graph [**15**]. The complexity of recognising subtree overlap graph is open. Because this problem is generally expected to be NP-complete, it becomes interesting to ask about reasonable subclasses where Gavril's algorithm could work, too.

With this in mind, we define three overlap subclasses of subtree overlap graphs: we define *k-SOG* as the *overlap graphs of subtrees in a tree such that the tree has at most k-leaves*, class *k-degree-POG* as the *overlap graphs of subpaths in a tree such that the tree has maximum degree at most k*, and the class *T-SOG* as the *overlap graphs of subtrees of a trees derived from an input tree T by subdivision of edges.*

Though we expect the recognition of subtree overlap graphs to be NP-complete, we expected the recognition of these simplified SOGs to be polynomial time. We were therefore surprised when we obtained hardness results for the recognition problems of k-SOG and k-degree-POG for a fixed integer $k \geq 3$ and for $T - SOG$ provided that T has at least three leaves. We present these hardness results in this paper. The result about k-degree-POG also holds for corresponding intersection graphs; our reduction also shows that it is NP-complete to recognise intersection graphs of paths in a tree with a fixed maximum degree greater than two. In contrast, intersection graphs of subpaths in a tree can be recognised in polynomial time.

Our result on the hardness of recognising the subtree overlap graph with k leafage for fixed integer $k \geq 3$ provides a counterpoint to work on the intersection leafage of chordal graphs. Stacho and Habib [**10**] give a polynomial-time algorithm for determining the leafage of a chordal graph and constructing a representation that achieves that leafage. Leafage was further explored by Chaplick and Stacho [**2**]

in terms of vertex-leafage where each subtree in the representation is permitted to have at most k leaves by showing that the vertex-leafage is polynomially solvable for vertex-leafage at most 3 and NP-complete otherwise. In contrast with the former and as another pebble into the mosaic of leafage, we show that determining the leafage of a subtree overlap graph is NP-hard for leafage at least 3.

Golumbic et al. [**9**] explore the complexity of recognising the intersection graphs of paths in a tree parameterised by both the maximum degree of the underlying tree and the number of vertices that must be shared between two paths for them to be considered as intersecting. They provide a complete hierarchy of graph classes using these parameters.

The idea of a simpler representation, as well as previous work by [**11**] motivated us to define complicacy for subtree overlap graphs: For a subtree overlap graph G, its *complicacy* is *minimum k, such that G is a k-SOG*. We denote this complicacy by $\mathrm{cmp}_S(G)$. For a natural number n, by $\mathrm{cmp}_S(n)$ we denote the minimum k such that every subtree-overlap graph on n vertices is also a k-SOG. Due to Cenek [**1**], it holds that $\mathrm{cmp}_S(n) \leq n$. As a further result, we obtained a lower bound $\mathrm{cmp}_S(n) \geq n - \log n + o(\log n)$.

1. Main ideas

In our classes, it is obvious that the recognition problem is in NP (there always exists a representation by subtrees of at most quadratic size with respect to the number of vertices). In order to show the hardness-results, we use a special version of the k-colouring problem, 3-connected k-colouring, i.e., the problem of colouring the graph with k colours so that no pair of neighbouring vertices gets the same colour, restricted only to 3-connected graphs. To observe that this problem is still NP-hard, it suffices to take an instance of the k-colourability problem. For a given graph, we take 3 independent copies of it and for a triple of vertices corresponding to one vertex, we put a triangle on these vertices, see Figure 1. Depending on the leafage of the underlying tree or the maximum permitted degree k of the underlying tree we reduce 3-connected k-colourability.

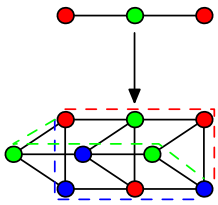

Figure 1. Idea of how to reduce (normal) k-colouring to 3-connected k-colouring. We reduce P_2. We also show, how to find three disjoint paths for one pair of vertices (dashed). Colours of the vertices correspond to the assigned colours. (in problem of colouring).

More thoroughly we proceed here for the recognition problem of subtree-overlap three graphs representable by subtrees of a tree with three leaves, as the remaining proofs use the same idea (just technical details and auxiliary constructions differ).

We define a *twig* in a tree to be a path from a leaf to the nearest vertex of degree higher than 2 (excluding the vertex of higher degree). For a tree with three leaves, there are three twigs that we obtain by removing the vertex of degree 3. The reduction exploits the fact that twigs in these trees can simulate colour-classes for an instance of our colouring problem. For a graph G, an instance of 3-connected 3-colourability, we define a graph G'' consisting of vertex-representatives of G, edge representatives (of G) and an overhead construction.

The overhead construction forces the vertex-representatives to be representable solely on twigs (that simulate colour-classes). Edge-representatives control individual pairs of incident vertices to be represented on two distinct twigs. In this way we obtain the assignment of colours for the original graph G from the representation of graph G''.

The overhead construction is generally represented by the G_d^u-blocking gadget depicted in Figure 2. Particularly when representing by subtrees in a tree with three leaves, we pick G_3^0. The subtree-overlap representation of this gadget generally blocks all branching-nodes in the underlying tree, as well, as all paths between them (i.e., between any pair of branching-nodes) in the case that the number of leaves is restricted.

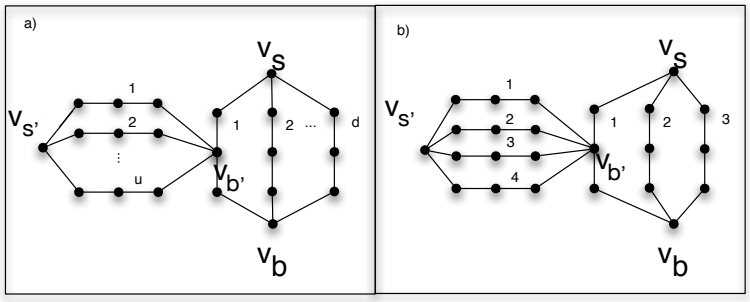

Figure 2. In a) the G_d^u graph – note the presence of d paths of three vertices between vertices v_s and v_b, and u paths of three vertices between $v_{b'}$ and $v_{s'}$. In b) an example: the G_3^4 graph.

Due to possible singularities in the representation that can turn up at most 4 times (in the whole construction), we pick 5 disjoint copies of the original graph (then at least one copy gets represented correctly). Vertex- and edge-representatives are designed in the following way: For each vertex v in G (i.e., in any of 5 copies of G), we make a pair of mutually adjacent vertices $v, f(v)$ in G''. We call these vertices a *representative* and its *brother*, respectively. For each edge $e = \{u, v\}$ in G, we put to G'' a vertex e adjacent to representatives of its endvertices, i.e., u and v. We make all *brothers* and *edge-representatives* adjacent with v_s and v_b from G_d^u. Moreover, we put a clique on them. This construction is illustrated by Figure 3. Vertex representatives are designed so that they can be represented (up to 4 exceptions) solely on twigs. Edge-representatives are designed so that they could not overlap a pair of representatives lying on the same twig.

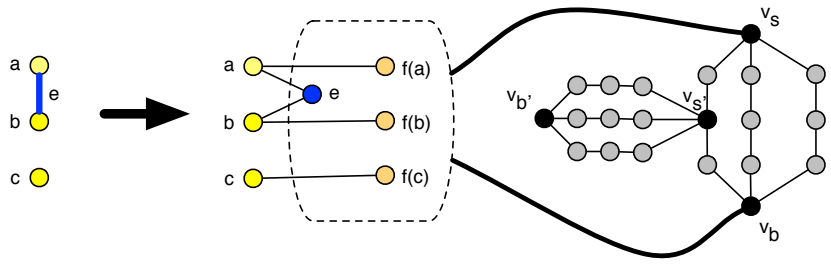

Figure 3. Demonstration of constructing G'' (on the right side of the arrow) from G (on the left side of the arrow). Yellow vertices (a, b, c) are vertex-representatives, peach vertices $(f(a), f(b), f(c))$ are their brothers and blue vertices are edge-representatives. Note that all vertices in the dotted oval induce a clique, and are all adjacent to v_b and v_s. In a full construction, there will be five copies of G involved, and therefore five copies of the vertex-representatives, brothers of vertices, and edge-representative. All edge-representative and brothers will induce a clique. These copies have been omitted for legibility.

Before proceeding further, let us define some notation. For a vertex v_a, by t_a we denote a subtree representing v_a. For a copy of G in G'' we say that in a subtree-overlap representation this copy is nicely represented, it all vertex representatives are represented on twigs and there is no illegal pair, i.e., two representatives of vertices adjacent in G represented on the same twig. A nice representation is therefore equivalent to a correct colouring of G. Obviously, a correct colouring admits a nice representation. The converse is more difficult.

Having described the graphs whose representability can be shown to be equivalent to colouring the original graph G, we state the appropriate theorems that follow from this construction.

2. Subtrees in restricted trees

The construction from the previous section shows the following.

Theorem 1. *For a given k, it is NP-complete to decide whether a given graph is a k-SOG.*

Proof. Sketch: We reduce 3-connected k-colouring and from its instance we create a graph G'' described in the previous section using G_d^u with $d = 3$ and if $k = 3$ then $u = 0$, otherwise $u = k-d+1$. It is not difficult to observe that a subtree representing v_s (a vertex of G_d^u) has to be contained in a subtree representing v_b, or vice-versa. Without loss of generality, we consider the former situation (mnemonics for vertices are v-smaller and v-bigger). Then this t_b has to be represented by a subtree containing all branching nodes in the underlying tree. From the following lemma (that gets shown in the full version of the article) it follows that at least one copy of the original graph G has to be nicely represented. □

Lemma 1. *When representing G'', no more than these singularities can occur: At most one vertex representative can be represented as a supertree of t_b. At most*

one vertex representative can be represented as a subtree of t_s containing a branching node. At most one vertex representative can be represented as a subtree of t_s on a non-twig path. At most one pair of neighbouring vertices can be represented as an illegal pair.

In a very similar way we obtain the following:

Theorem 2. *Given a tree T, it is NP-hard to decide for a given graph G whether it is a T-SOG.*

Sketch of proof. As T has k leaves, we reduce 3-connected k-colouring in a described way. In a tree, a vertex of degree d is called a lastbranch, if it is incident with $d-1$ twigs. Let d be a minimum degree of a lastbranch in T. Then for the construction we pick G_d^{k-d+1} and the same arguments apply. □

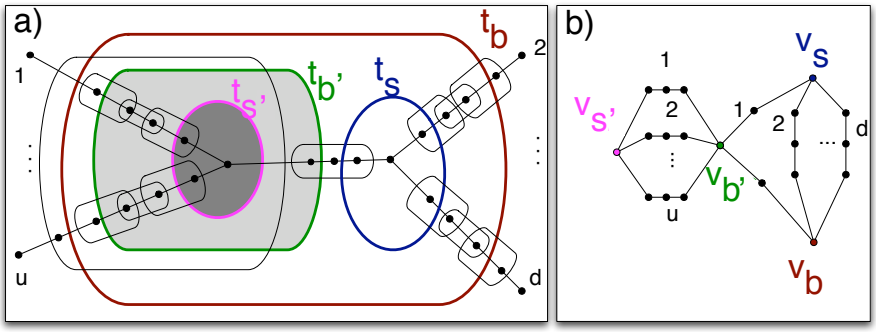

Figure 4. A generalised overlap representation of the G_d^u graph on a tree with a node such that the forest created by removing that node has two connected components: a tree with d leaves and a node of degree d and a tree with at least $u+1$ leaves. The interior of $t_{b'}$ and $t_{s'}$ are darkened to indicate that the structure of the tree there is somewhat irrelevant - only the number of boundary nodes is important. There is exactly one node of degree greater than two contained in t_s and that node is contained only in t_s and t_b, and all other nodes of degree greater than two are contained in $t_{b'}$. The representation is on the left, and the G_d^u graph is on the right. Vertex labels and corresponding subtrees are colour coded.

3. Paths in a Tree

We now turn to the complexity of the problem of recognising overlap graphs of paths in a tree of maximum degree k.

Theorem 3. *For a given $k > 2$, it is NP-complete to decide whether a given graph G has an overlap representation by subpaths in a tree with maximum degree k. To decide whether G has an intersection representation by subpaths in a tree with maximum degree k, is NP-complete, too.*

Sketch of proof. This time we use G'' without G_d^u, i.e., without all its vertices (just vertex representatives, their brothers and edge representatives remain). In

the representation, there appears a *central vertex*, i.a. a branching node used by all edge representatives (otherwise they could not mutually overlap). Removal of this vertex splits the tree into k components corresponding to colour-classes for G. At most two vertex representatives can contain the central vertex, remaining copies of G have to be nicely represented. □

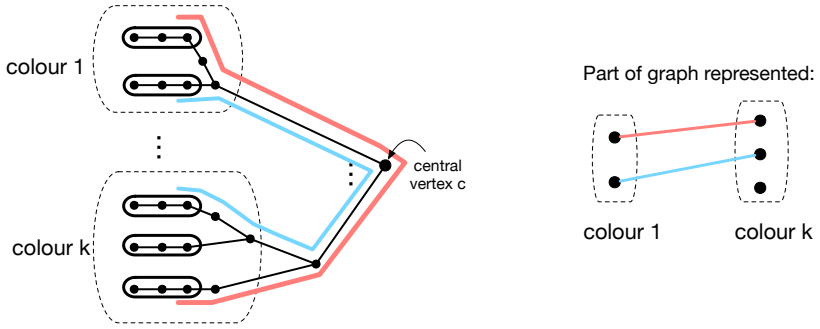

Figure 5. Sample subpath overlap representation on a tree of maximum degree $k \geq 3$. Black ovals are vertex-representatives, the two pale coloured subtrees are edge-representatives. Brothers of vertex-representatives are omitted. Colour classes are shown in dotted ovals. The part of the graph that is represented is shown on the right.

4. Conclusion and future work

We have shown that recognising a number of subclasses of subtree overlap graphs is NP-complete. This is surprising because of the extreme enforced simplicity of the representations. Our ultimate goal continues to be resolving the complexity of the recognition problem for subtree overlap graphs in general. This is currently an open problem.

Other related open problems include tighter bounds on the complicacy of subtree overlap graphs, as well as investigation of other geometric overlap and intersection classes. Also a challenging problem related to subtree overlap graphs (a.k.a. subtree filament graphs) is a weighted clique (as Gavril's algorithm a bit surprisingly covers just maximum weighted independent set in this case).

Further open problems reflect the approximability of presented parameters, i.e., whether for a graph representable in a tree with k leaves we can efficiently find a representation in a tree with kl leaves for some l and similarly with subpaths in a tree if we permit a higher maximum degree in the underlying tree than necessary.

References

1. Cenek E., *Subtree Overlap Graphs and the Maximum Independent Set Problem*, Master's thesis, University of Alberta, Department of Computing Science, 1998.

2. Chaplick S. and Stacho J., *The vertex leafage of chordal graphs*, Discrete Appl. Math. **168** (2014), 14–25.

3. Corneil D. G., Olariu S. and Stewart L., *Lbfs orderings and cocomparability graphs*, in: SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 883–884.
4. Dangelmayr C. and Felsner S., *Chordal graphs as intersection graphs of pseudosegments*, in: Graph Drawing (M. Kaufmann and D. Wagner, eds.), Lecture Notes in Comput. Sci. 4372, Springer, 208–219.
5. Enright J. and Stewart L., *Subtree filament graphs are subtree overlap graphs*, Inf. Process. Lett. **104** (2007), 228–232.
6. Gavril F., *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combin. Theory Ser. B **16** (1974), 47–56.
7. Gavril F., *Maximum weight independent sets and cliques in intersection graphs of filaments*, Inf. Process. Lett. **73** (2000), 181–188.
8. Gilmore P. and Hoffman A., *A characterization of comparability graphs and of interval graphs*, Canad. J. Math. **16** (1964), 539–548.
9. Golumbic M. C., Lipshteyn M. and Stern M., *Equivalences and the complete hierarchy of intersection graphs of paths in a tree*, Discrete Appl. Math. **156** (2008), 3203–3215.
10. Habib M. and Stacho J., *Linear algorithms for chordal graphs of bounded directed vertex leafage*, Electron. Notes Discrete Math. **32** (2009), 99–108.
11. Kratochvíl J. and Pergel M., *Two results on intersection graphs of polygons*, in: Graph Drawing (G. Liotta, ed.), Lecture Notes in Comput. Sci. 2912, Springer, 59–70.
12. Monma C. L. and Wei V. K.-W., *Intersection graphs of paths in a tree*, J. Comb. Theory, Ser. B **41** (1986), 141–181.
13. Pergel M., *Recognition of polygon-circle graphs and graphs of interval filaments is np-complete*, in: WG 2007 (A. Brandstädt, D. Kratsch, H. Müller, eds.), Lecture Notes in Comput. Sci. 4769, Springer, 238–247.
14. Rose D. J., Tarjan R. E. and Leuker G. S., *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput. **5** (1976), 266–283.
15. Schäffer A. A., *A faster algorithm to recognize undirected path graphs*, Discrete Appl. Math. **43** (1993), 261–295.

J. Enright, University of Edinburgh, Easter Bush, Midlothian, UK,
e-mail: jessica.enright@ed.ac.uk

M. Pergel, Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic,
e-mail: perm@kam.mff.cuni.cz