



Thick Forests

Martin Dyer¹, Haiko Müller^{*,1}

School of Computing, University of Leeds, Leeds LS2 9JT, UK

ARTICLE INFO

Article history:

Received 3 March 2025

Received in revised form 16 December 2025

Accepted 16 December 2025

Keywords:

Counting

FPRAS

Unipolar graph

ABSTRACT

We consider classes of graphs, which we call *thick graphs*, that have the vertices of a corresponding *thin* graph replaced by cliques and the edges replaced by cobipartite graphs. In particular, we consider the case of *thick forests*, which we show to be the largest class of perfect thick graphs.

Recognising membership of a class of thick $\mathcal{C}$ -graphs is NP-complete unless the class $\mathcal{C}$ is triangle-free, so we focus on this case. Even then membership can be NP-complete. However, we show that the class of thick forests can be recognised in polynomial time.

We consider two well-studied combinatorial problems on thick graphs, independent sets and proper colourings. Since determining the independence or chromatic number of a perfect graph is known to be tractable, we examine the complexity of *counting* all independent sets and colourings in thick forests.

Finally, we consider two parametric extensions to larger classes of thick graphs: where the parameter is the size of the thin graph, and where the parameter is its treewidth.

© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

We introduce classes of graph, which we call *thick graphs*, that can be represented with cliques as vertices and connecting cobipartite graphs as edges. We call the cliques *thick vertices*, or *nodes* and the cobipartite graphs *thick edges*, or *links*. The graph given by shrinking the cliques to single vertices will be called the *thin graph*, which is usually far from unique. The purpose is to simplify the input graph structure in order to facilitate algorithmic applications.

More formal definitions are given in Section 2. General notation and definitions are given in Section 1.2.

In particular, we are interested in whether a given graph has any thin graph lying in some particular graph class. This is the *recognition* problem. However, it seems to be NP-complete for most classes of thin graphs, for example bipartite graphs, as we show in Section 2.2. We also consider the recognition problem when the input graph must itself lie in some given graph class. We give formal definitions in Section 2.

We show in Section 2.2 that only thick triangle-free graphs can be easily recognised, with a shorter proof than one previously given in [37]. However, since it is NP-complete to recognise thick bipartite graphs, not all triangle-free thin graphs are recognisable in P. For the triangle-free case, a fixed-parameter (XP) recognition algorithm was given [37], when the parameter is the thin graph H . But this did not show fixed-parameter tractability (FPT). In Section 7.1, we use the methods developed in Section 4 to give an FPT algorithm for this class, when the parameter is the number of vertices in the thin graph H . We also show that this problem is computationally equivalent to that considered in [37].

* Corresponding author.

E-mail addresses: m.e.dyer@leeds.ac.uk (M. Dyer), h.muller@leeds.ac.uk (H. Müller).

¹ Work supported by EPSRC, United Kingdom grant EP/S016562/1 “Sampling in hereditary classes”.

Since most thick classes lead to NP-complete recognition problems, we focus on a potentially tractable case, where the thin graph must be a *forest*. We show in Section 3 that thick forests are perfect graphs, resembling chordal graphs in some respects. Whereas chordal graphs have clique trees, thick trees are trees of cliques. However, though it is obvious that thick forests are not chordal in general, we show that not all chordal graphs are thick forests. We develop a polynomial time recognition algorithm for thick forests in Section 4. This forms the central section of the paper.

The union of chordal graphs and thick forests is (properly) contained in a larger class $\mathcal{Q}$ of graphs, which we call *quasi* thick forests, defined in Section 5. We show in Section 5 that $\mathcal{Q}$ is a class of perfect graphs, implying the same result for thick forests, which we give in Section 3.1. These can be recognised efficiently by *clique cutset decomposition* [42], though this does not give efficient recognition for thick forests, as it does for recognition of chordal graphs. Nevertheless, it is more suited to algorithmic applications.

We consider two classical graph problems which are intrinsically related to perfection: independent sets and colourings. That the decision version of these problems is in P for $\mathcal{Q}$ follows from perfection, though we note that the known algorithms are not combinatorial and require continuous optimisation methods. Therefore we consider the counting versions which are hard for general perfect graphs. Counting colourings and independent sets *exactly* are both #P-complete for perfect graphs, even for bipartite graphs, and their *approximate* versions are both #BIS-hard [17]. So we are interested in classes of graphs for which polynomial time counting is possible.

We show in Section 6.2 that counting colourings exactly remains #P-complete for thick forests, but approximate counting is in polynomial time for $\mathcal{Q}$. We use an approach introduced by Dyer, Jerrum, Müller and Vučković [20] for approximately counting independent sets in claw-free perfect graphs via clique cutset decomposition. We show here that the technique of [20] yields polynomial time approximately counting of colourings of graphs in $\mathcal{Q}$, though this was not possible for claw-free perfect graphs. Note that thick forests are far from claw-free, since they contain all trees. In Section 6.3, we show further that even approximate colouring is not possible in any class of thick graphs which is not a subset of thick forests.

By contrast, we show in Section 6.1 that exact counting of (weighted) independent sets can be done in polynomial time for $\mathcal{Q}$, again using the technique of [17], so the counting problem is little harder than for chordal graphs.

Since forests are precisely the graphs of *treewidth* 1, and thick treewidth-1 graphs are tractable, in Section 7.2 we consider thick treewidth- k graphs, with $k > 1$ a small constant. We regard this as a parameterisation of the thin graph, where the parameter is treewidth. We show that recognition of this class is NP-complete for all $k > 1$, so we are obliged to assume that a solution of the recognition problem is supplied. That is we are given a thin graph of treewidth at most k . But even then, we show that counting colourings is hard, even approximately. However, on the positive side, we give an XP algorithm for exactly counting independent sets.

1.1. Previous work

Thick graphs appear to have been explicitly introduced by Salamon and Jeavons [40] in the context of the Constraint Satisfaction Problem (CSP). The satisfiability problem is then equivalent to finding the independence number of a thick graph where the thin graph is in some class. In particular, the class of thick trees was considered in [40], since the resulting tree-structured CSP can be solved efficiently.

Thick trees have also been studied as *unipolar* graphs, e.g. [22,38,43]. In our terminology, these are *thick stars*. So G is a unipolar graph if it has nodes comprising a *hub* h and leaves $b_1, b_2, \dots, b_s$, the *satellites* and only the links hb_i ($i \in [s]$). Polynomial time algorithms for the recognition problem have been given in [22,38,43]. We give another in Section 4.2 below. Since a star is a tree with at most one interior vertex, these are a subclass of thick forests.

More recently, Kanj, Komusiewicz, Sorge and van Leeuwen [34] have generalised this to give an FPT (fixed-parameter tractable) algorithm for recognising thick bipartite graphs using a bound on the size of one of the parts as the parameter. Thus these graphs have a “hub” which is a cluster graph of fixed size.

The thick graph concept is implicit in *subcolourings* of graphs, introduced by Albertson, Jamison, Hedetniemi and Locke [2]. A q -subcolourable graph is precisely a thick q -colourable graph. The recognition problem is to decide whether G has any q -colourable thin graph H . In particular, 2-subcolourability is the problem of recognising a thick bipartite graph. Subcolourability has generated a literature of its own. See, for example, [9,27].

Stacho [41] considered 2-subcolourable chordal graphs. We show in Section 3.2 that these are precisely chordal thick forests. Stacho gave a polynomial time recognition algorithm for this case, but his approach does not extend to general thick forests. We show in Section 4 that all thick forests can be recognised in P, which implies Stacho’s result as a special case.

The case where the thin graph is a fixed graph H was called an (H, K) -partition of the input graph G by MacGillivray and Yu [37]. As mentioned above, they gave an almost complete solution to the recognition problem in this case. They showed that the recognition problem is NP-complete unless the thin graph is triangle-free. We review and improve their work in Section 7.

A similar question to that of [37] is that of *matrix partitions*, e.g. [24,25]. A fixed symmetric matrix H is given which determines the structure of valid partitions. If the matrix is the adjacency matrix of H but with 1’s on the diagonal, 0’s for the non-edges and *’s for the edges, then an H -partition of G is precisely an (H, K) -partition of G .

1.2. Preliminaries

We summarise our notation and definitions. Most readers will be able to use this section purely for reference.

$\mathbb{N}$ will denote the set of positive integers. We write $[m] = \{1, 2, \dots, m\}$ for any $m \in \mathbb{N}$. For any integers $a \geq b \geq 0$, we write $(a)_b = a(a-1) \cdots (a-b+1)$ for the *falling factorial*, where $(a)_0 = 1$ for all $a \geq 0$, and $(a)_b = 0$ if $b > a$ or $b < 0$.

Throughout this paper, all graphs are simple finite and undirected. For a graph $G = (V, E)$ we set $V(G) = V$, $E(G) = E$, with $n = |V|$ and $m = |E|$. If $v \in V(G)$ or $e \in E(G)$ we may simply write $v \in G$ or $e \in G$. We will write $G \cong H$ if G is isomorphic to H . If $e = \{u, v\}$ for $u, v \in V$, we will write $e = uv$. A *clique* is a graph (V, E) having all possible edges, i.e. $E = V^{(2)}$ where $V^{(2)} = \{vw : v, w \in V, v \neq w\}$. A clique is *maximal* if it is not a proper subgraph of any clique in G . If C is a clique and A is a maximal clique containing it, we will say that we can *expand* C to A .

We use $A \uplus B$ to denote the disjoint union of sets A, B . The disjoint union of graphs G_1, G_2 is then $G_1 \uplus G_2 = (V_1 \uplus V_2, E_1 \uplus E_2)$. A *cluster graph* is a disjoint union of cliques.

The *complement* $\bar{G} = (V, \bar{E})$ has $vw \in \bar{E}$ if and only if $v \neq w$ and $vw \notin E$. G is a clique if and only if $\bar{G}$ is an *independent set*. If $G \in \mathcal{B}$, the class of bipartite graphs, then $\bar{G}$ is cobipartite, two cliques joined by the edges of a bipartite graph.

If $B = (V_1 \cup V_2, E)$ is bipartite, its *bipartite complement* is the graph $\bar{B} = (V_1 \cup V_2, \bar{E})$, where $\bar{E} = \{v_1v_2 : v_1 \in V_1, v_2 \in V_2, v_1v_2 \notin E\}$.

We say V_1 is *complete* to V_2 in G if $v_1v_2 \in E$ for all $v_i \in V_i$ ($i \in [2]$), and *anticomplete* to V_2 if $v_1v_2 \notin E$ for all $v_i \in V_i$ ($i \in [2]$). These relations are symmetrical between V_1 and V_2 .

The term *induced subgraph* will mean a vertex-induced subgraph, and the subgraph of $G = (V, E)$ induced by the set S will be denoted by $G[S]$. Where safe we will not distinguish between a set of vertices and the subgraph it induces. A subgraph $G[S]$ is a *component* of G if it is connected and there is no edge from S to $V \setminus S$. If $G[S] \cong H$ for some $S \subseteq V$, we say G contains H . A forbidden (induced) subgraph for G is a graph H such that G does not contain H .

A class of graphs $\mathcal{C}$ is *hereditary* if for any $G = (V, E) \in \mathcal{C}$ and any $S \subseteq V$, we have $G[S] \in \mathcal{C}$. A class $\mathcal{C}$ is *additive* (or *union-closed*) if a graph is in $\mathcal{C}$ whenever all its connected components are in $\mathcal{C}$.

The *neighbourhood* $\{u \in V : uv \in E\}$ of $v \in V$ will be denoted by $N(v)$, and the *closed neighbourhood* $N(v) \cup \{v\}$ by $N[v]$. The neighbourhood $N(S)$ of a set $S \subseteq V$ is the set $\bigcup_{v \in S} N(v) \setminus S$, and $N[S] = \bigcup_{v \in S} N[v]$. We write $\deg(v)$ for the degree $|N(v)|$ of v in G .

Two vertices $u, v \in V$ are called *false twins* if $N(u) = N(v)$ and *true twins* if $N[u] = N[v]$.

If $\emptyset \subset V_1, V_2 \subset V$, with $V_1 \cap V_2 = \emptyset$, we denote the edge set $\{vw \in E : v \in V_1, w \in V_2\}$ by $V_1 : V_2$. Then $V_1 : V_2$ is a *cut* in G if $V = V_1 \uplus V_2$.

A set $S \subseteq V$ is a *cutset* or *separator* of G if, for some $\emptyset \neq V_1, V_2 \subseteq V$, we have $V = V_1 \cup V_2$, $V_1 \cap V_2 = S$, and there is no edge $vw \in E$ with $v \in V_1 \setminus S$ and $w \in V_2 \setminus S$. If $G[S]$ is a complete graph (clique), then S is a *clique separator*.

A set $S \subseteq V$ is an *independent set* if we have $vw \notin E$ for all $v, w \in S$. The maximum size of an independent set in G is denoted by $\alpha(G)$, the *independence number* of G . We consider the problem of counting all independent sets in G , and we will denote this number by $\#ind(G)$. The size of the largest clique in G is denoted $\omega(G)$, so $\omega(G) = \alpha(\bar{G})$. A *matching* $M \subseteq E$ in G is a set of disjoint edges. That is, $e_1 \cap e_2 = \emptyset$ for all $e_1, e_2 \in M$.

A *q-colouring* ψ of the vertices of a graph G , with colour set Q of size q , is a function $\psi : V \rightarrow Q$. The colour classes are the sets $\psi^{-1}(i)$ for $i \in Q$. It is a *proper colouring* if $\psi(u) \neq \psi(v)$ for all $uv \in E$, that is, all colour classes are independent sets. Here $\psi(S) = \{\psi(v) : v \in S\}$ for $S \subset V$. By convention, the adjective “proper” is omitted if there is no ambiguity. We consider the problem of counting (proper) q -colourings of (the vertices) of G . Denote this number by $\#col_q(G)$. The corresponding decision problem $\exists col_q(G)$ asks $\#col_q(G) > 0$? The minimum q for which this is true is the *chromatic number* $\chi(G)$ of G . A graph is a clique if $\chi(\bar{G}) = 1$. It is *bipartite* if $\chi(G) \leq 2$, and its complement $\bar{G}$ is *cobipartite*.

A graph $G = (V, E)$ is *perfect* if $\chi(G[S]) = \omega(G[S])$ for all $S \subseteq V$. A (k) -hole $G[S]$ in a graph is an induced subgraph isomorphic to a cycle C_k of length $k \geq 4$, a *long hole* if $k \geq 5$. (A 4-hole is often called a *square*). A (k) -antihole is the complement $\bar{C}_k$ of a hole. Note that a 4-antihole $\bar{C}_4 = 2K_2$ is disconnected, so antihole usually means the complement of a long hole, that is, a long antihole. The *strong perfect graph theorem* (SPGT) [11] shows that G is perfect if and only if it contains no induced hole or antihole with an odd number of vertices. This clearly implies the earlier (weak) *perfect graph theorem* (PGT) [36], that $\bar{G}$ is perfect if G is perfect. It is easy to see that a bipartite graph is perfect, and hence a cobipartite graph G is perfect by the PGT. In perfect graphs, a q -colouring and a maximum independent set can be found in polynomial time [29], though only by an indirect optimisation algorithm. Exactly counting independent sets or q -colourings is $\#P$ -complete in general [18] and approximate counting is $\#BIS$ -hard [17], even for the class of bipartite graphs. Thus positive counting results are possible only for restricted classes of perfect graphs.

A graph is *chordal* (or *triangulated*) if it has no hole. It is easy to see that it can have no long antihole: A 5-antihole is also a 5-hole, so forbidden. A hole of size 6 or more contains a $2K_2$, so its complement contains a 4-hole and is forbidden. Thus chordal graphs are perfect. A chordal graph can be recognised by finding a *perfect elimination order* of its vertices. See [39], which gives a linear time algorithm for constructing the ordering.

A graph is *chordal bipartite* if it is bipartite and has no long holes. Note that these graphs are not chordal in general, despite the name.

Chain graphs [45] form the class of graphs which exclude triangles, 4-antiholes and 5-holes. Since a chain graph excludes $2K_2$, it excludes any hole or antihole of size greater than five, or path of more than three edges. Its components are connected bipartite graph with nested neighbourhoods. The complement of a chain graph is a cobipartite chordal graph, called a *cochain graph*.

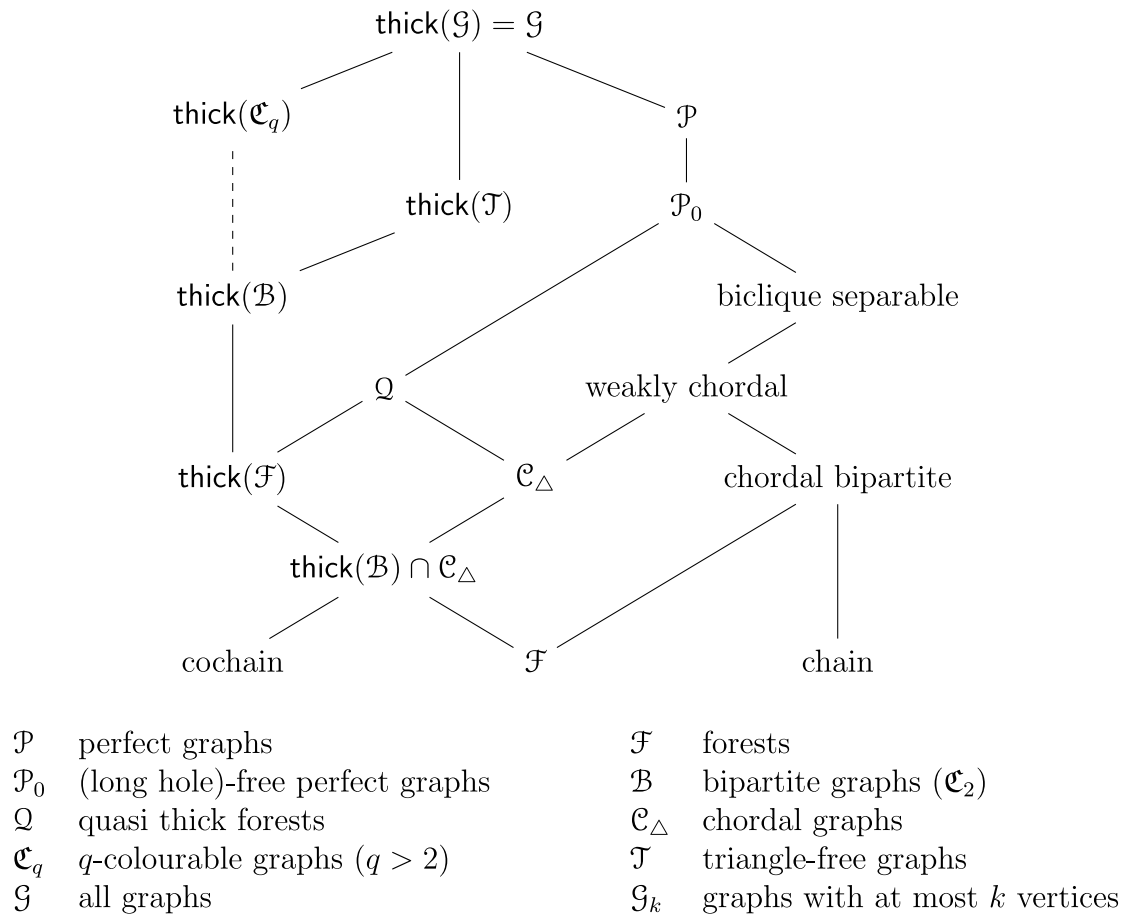


Fig. 1. Graph classes and their inclusions.

For graph theoretic background not given above, see Diestel [16], for example.

Finally, we use standard terms from computational complexity without comment. For further information see, for example, Bovet and Crescenzi [8] for a general introduction, Jerrum [31] for counting complexity and Fellows [26] for parameterised complexity.

1.2.1. Graph classes

We will consider several related graph classes, so we give an indication of their relationship and our (nonstandard) notation in Fig. 1. Here an upward line denotes strict inclusion, where the non-obvious inclusions will be justified later. Most of these classes are well studied. The main exceptions, $\text{thick}(\mathcal{F})$ and $\mathcal{Q}$, are the topic of this paper.

1.2.2. Treewidth

A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, b) where $T = (I, A)$ is a tree and b maps the nodes of T to subsets of V such that

- (1) for every vertex $v \in V$ exists a node $i \in I$ such that $v \in b(i)$,
- (2) for every edge $e \in E$ exists a node $i \in I$ such that $e \subseteq b(i)$, and
- (3) for every vertex $v \in V$ the graph $T[\{i \mid v \in b(i)\}]$ is connected.

Clearly, the connected graph in the third property is a subtree of T . For a node $i \in I$, the set $b(i) \subseteq V$ is called a *bag* of the tree decomposition.

Every graph $G = (V, E)$ has a tree decomposition with just one bag containing all vertices. That is, $I = \{i\}$, $A = \emptyset$ and $b(i) = V$. If G is a tree then we can choose a tree T that is a subdivision of G . Assuming $V \cap E = \emptyset$, this can be formalised to $I = V \cup E$, $A = \{ve \mid v \in e, e \in E\}$, $b(v) = \{v\}$ for all $v \in V$ and $b(e) = e$ for all $e \in E$.

The *width* of (I, b) is the maximum size of a bag $b(i)$ minus one, where $i \in I$. The *treewidth* of G , denoted by $\text{tw}(G)$, is the minimum width of a tree decomposition of G .

For a clique K_n on n vertices, the tree decomposition with a single bag is optimal, so $\text{tw}(K_n) = n - 1$. For a tree G with at least one edge, the decomposition mentioned above is optimal, so $\text{tw}(G) = 1$.

For every graph $G = (V, E)$ exists a tree decomposition (T, b) of minimum width such that $|I| \leq |V|$, where $T = (I, A)$ as above. Such a (T, b) can be obtained from a tree decomposition of width $\text{tw}(G)$ by contracting arcs $ij \in A$ with $b(i) \subseteq b(j)$. In general, contracting an arc $ij \in A$ (and assigning the bag $b(i) \cup b(j)$ to the resulting vertex) leads to another tree decomposition of G , possibly of larger width.

By the inverse operation of splitting nodes we can ensure, for every arc $ij \in A$, that $b(i)$ and $b(j)$ differ by at most one vertex. Formally, a tree decomposition (T, b) is *nice* if the tree $T = (I, A)$ has a root $r \in I$ with $b(r) = \emptyset$ such that every node $i \in I$ is of one of the following types:

leaf For every leaf l of T we have $b(l) = \emptyset$.

introduce An introduce node i has exactly one child j , $b(i) \setminus b(j)$ is a singleton and $b(j) \subset b(i)$ holds.

forget A forget node i has exactly one child j , $b(j) \setminus b(i)$ is a singleton and $b(i) \subset b(j)$.

join A join node i has exactly two children j and k such that $b(i) = b(j)$ and $b(i) = b(k)$ hold.

A slightly less restricted version of nice tree decompositions was given in [35]. Every graph $G = (V, E)$ has a nice tree decomposition (T, b) of minimum width such that $|I| = \mathcal{O}(|V|)$.

Computing treewidth is NP-hard, even when restricted to co-bipartite graphs [3]. There is an $\mathcal{O}(n^{k+2})$ XP algorithm [3] for obtaining a tree decomposition of width at most k , if one exists, and also a linear time FPT-algorithm [5]. For further information on treewidth see [6].

2. Thick graphs

A *thick vertex*, or *node*, is a clique, and a *thick edge*, or *link*, is a cobipartite graph. Informally, a *thick graph* is a graph with nodes and links. Where we use these terms simultaneously, it will imply that vertex $\subseteq$ node $\subseteq$ thick vertex and edge $\subseteq$ link $\subseteq$ thick edge.

Formally, we define a thick graph as follows. Let $G = (V, E)$ and $H = (\mathcal{V}, \mathcal{E})$ be graphs, where $n = |V|$, $v = |\mathcal{V}|$. A surjective function $\psi : V \rightarrow \mathcal{V}$ such that

1. for all $v \in \mathcal{V}$ the set $\psi^{-1}(v) \subseteq V$ is a clique in G , and
2. for all $uv \in E$ we have $\psi(u) = \psi(v)$ or $\psi(u)\psi(v) \in \mathcal{E}$,

will be called a *model* (H, ψ) of G . Then G is the *thick graph* and H is the *thin graph*. Intuitively, G is given by identifying each vertex in $\mathbf{u} \in \mathcal{V}$ with a node (clique) $\mathbf{u} \subseteq V$ and each edge in $\mathcal{E}$ with a link (cobipartite graph) $\mathbf{vw}$ in G .

If $\mathbf{u} \in \mathcal{V}$ then $\psi^{-1}(\mathbf{u})$ is a clique in G , an r -clique $\{u_1, u_2, \dots, u_r\}$ in G , we will identify the node $\mathbf{u}$ with this clique, so $G[\mathbf{u}] \cong K_r$. Similarly, if $\mathbf{vw} \in \mathcal{E}$, the link $\mathbf{vw}$ will be identified with the cobipartite graph $G[\mathbf{u} \cup \mathbf{w}]$. Thus we will omit the term “thick” if it is clear from the context.

We will write $\text{thick}(H) = \{G : G \text{ has a model } (H, \psi)\}$ and $\text{thin}(G) = \{H : G \text{ has a model } (H, \psi)\}$. Clearly $H \in \text{thick}(H)$ and $G \in \text{thin}(G)$, by taking $\psi = \text{id}$, the identity map.

Clearly $\text{thin}(G)$ corresponds to the set of *clique covers* of G . For G an independent set, $|\text{thin}(G)| = 1$. For G an n -clique, $|\text{thin}(G)| = B_n$, the n th Bell number. For triangle-free G , $|\text{thin}(G)|$ is the number of matchings in G . Thus the size of $\text{thin}(G)$ is a #P-complete quantity in general [44]. On the other hand, $\text{thick}(H)$ is always countably infinite unless H is the empty graph $(\emptyset, \emptyset)$.

The *clique cover number* of G is $\theta(G) = \chi(\bar{G}) = \min\{|\mathcal{V}| : H \in \text{thin}(G)\}$. So asking if $\theta(G) \leq k$ is equivalent to $G \in \text{thick}(\mathcal{G}_k)$, where $\mathcal{G}_k$ is the class of graphs with at most k vertices.

If $\mathcal{C}$ is any graph class, we will write the class $\bigcup_{H \in \mathcal{C}} \text{thick}(H)$ as $\text{thick}(\mathcal{C})$. This class is hereditary if $\mathcal{C}$ is hereditary, since cluster graphs are hereditary. Thus $\text{thick}(\cdot)$ is a heredity-preserving operator on graph classes.

It should be observed that graphs in $\text{thick}(\mathcal{C})$ may have very different properties from graphs in $\mathcal{C}$. For example, bipartite graphs are perfect, but a thick bipartite graph is not necessarily perfect, see Fig. 4. More obviously, a tree has no holes, but even a thick edge can have $\mathcal{O}(n^4)$ 4-holes.

Thus $G \in \text{thick}(\mathcal{C})$ if there is any $H \in \mathcal{C}$ such that $G \in \text{thick}(H)$. For given G and fixed $\mathcal{C}$, the problem of deciding if $G \in \text{thick}(\mathcal{C})$ is the *recognition* problem for $\text{thick}(\mathcal{C})$. More generally, given two hereditary classes $\mathcal{C}_1, \mathcal{C}_2$ and $G \in \mathcal{C}_1$, we might ask whether there is any $H \in \mathcal{C}_2$ such that $G \in \text{thin}(H)$. This is the recognition problem for the class $\text{thick}(\mathcal{C}_2) \cap \mathcal{C}_1$, which we will write as $\text{thick}(\mathcal{C}_2|\mathcal{C}_1)$. When $\mathcal{C}_1$ is the class $\mathcal{G}$ of all graphs, this is simply $\text{thick}(\mathcal{C}_2)$. When $\mathcal{C}_2$ is $\mathcal{G}$ this is $\mathcal{C}_1$. The class of thick graphs which are in $\mathcal{C}_1$ is precisely $\mathcal{C}_1$, because $G \in \text{thick}(G)$. Equivalently, we ask whether there is any $H \in \text{thin}(G)$ for $G \in \mathcal{C}_2$ such that $H \in \mathcal{C}_1$. In general, this problem is NP-complete. Let $\mathcal{T}$ be the class of triangle-free graphs.

Lemma 1. Let $\mathcal{C}$ be any class such that $\mathcal{C} \not\subseteq \mathcal{T}$, then the recognition problem for $\text{thick}(\mathcal{C})$ is NP-complete.

Proof. For the sake of a contradiction assume some $H = (\mathcal{V}, \mathcal{E}) \in \mathcal{C}$ contains a triangle $T = (abc)$. We construct G as follows. Let $G_0 = (V_0, E_0)$ be an instance of recognising $G_0 \in \text{thick}(T)$. This is NP-complete since it is 3-colourability of $\bar{G}_0$. We may assume G_0 contains an independent set $\{a_0, b_0, c_0\}$ of size 3. Otherwise $\chi(\bar{G}_0) \leq 2$ so $\bar{G}_0$ is bipartite, and hence G_0 is cobipartite, and recognition is in P , see Section 4.2.

To construct $G = (V, E)$ we first replace T by G_0 , and for each $v \in \mathcal{V} \setminus \{a, b, c\}$, we set $v \in V$. If $vw \in \mathcal{E}$, then if $w \notin \{a, b, c\}$ set $vw \in E$, and if $w \in \{a, b, c\}$, set $vw_0 \in E$. Then G_0 has a model (T, ψ_0) if and only if G has a model (H, ψ) with $\psi(v) = \psi_0(v)$ for $v \in V_0$ and $\psi(v) = v$ for $v \in V \setminus V_0$. Thus deciding $G \in \text{thick}(H)$ is equivalent to deciding $G_0 \in \text{thick}(T)$, so is NP-complete. $\square$

Thus tractable recognition is only possible for classes of triangle-free thin graphs.

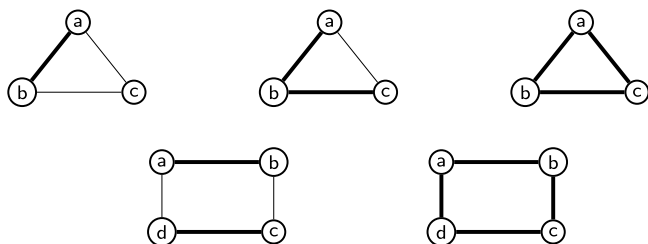


Fig. 2. Loose triangles and 4-cycles.

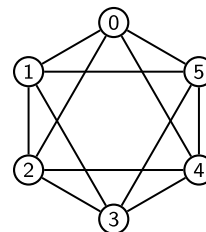


Fig. 3. Full links.

2.1. Full links and full graphs

A *full link* is a link which is a clique. Equivalently, the cut between its end nodes is a complete bipartite graph. These can be problematic since they can also be viewed as thick vertices. For a thick triangle, if any edge is full, this may be viewed as a thick edge. Consider the triangle abc in Fig. 2. The heavy lines represent full links, the other solid lines non-full but nonempty links, and the dashed lines non-full but possibly empty links. If ab is the only full link then the triangle (abc) is the thick edge $ab : c$. If both ab and bc are thick edges, the triangle can be either of the two thick edges, $ab : c$ or $a : bc$. If all three ab , bc and ac are full then the triangle can be any of three thick edges $ab : c$, $a : bc$, $ac : b$, or the thick vertex abc . Thus there can be a choice in how the triangle is modelled. We will call these *loose triangles*, and we will not model them as thick triangles. A thick triangle with no full links will be called a *stiff triangle*. The possible ambiguity in loose triangles causes a difficulty in the recognition of thick trees in Section 4. We will consider this in Section 4.4.

We might also have a 4-cycle $(abcd)$ as in Fig. 2. This represents a thick edge if and only if either of the pairs of non-adjacent edges ab, cd or ac, bd are both full. If all four edges are full, there is a choice of representation as a thick edge. We will call these loose 4-cycles, and any other a *stiff 4-cycle*. These 4-cycles cannot have chords, or the 4-cycle is two thick triangles with a common link.

Note that, if G contains a stiff triangle or 4-cycle, we can reject it if we are trying to recognise $G \in \text{thick}(\mathcal{F})$.

We will call a thick graph which has only full links a *full graph*. These form a subclass of thick graphs. They can clearly have no stiff triangles. So it might seem from the above that we can assume that full graphs are triangle-free, but this is not so. Reducing a full triangle to a thick edge or vertex will generally lead to a graph which is no longer full.

To illustrate, consider the graph in Fig. 3. The only model as a full graph is the identity. It has two models as a stiff triangle with nodes $\{0, 1\}$, $\{2, 3\}$, $\{4, 5\}$ and $\{1, 2\}$, $\{3, 4\}$, $\{5, 0\}$ respectively. It has six models as a thick P_2 , with the nodes $\{i, i+1, i+2\}$, $\{i+3, i+4, i+5\} \pmod{6}$ for $i \in [6]$.

Let $\text{full}(H)$ denote the class of graphs obtained by replacing the vertices of the graph H with cliques and its edges by full links. Then, if $\mathcal{C}$ is any graph class, let $\text{full}(\mathcal{C}) = \bigcup_{H \in \mathcal{C}} \text{full}(H)$. Thus $\text{full}(\cdot)$ is again an operator on graph classes. Clearly $\text{full}(\mathcal{C}) \subset \text{thick}(\mathcal{C})$ for any $\mathcal{C}$.

In general, graphs in $\text{thick}(\mathcal{C})$ usually have very different properties from graphs in $\mathcal{C}$. For example, bipartite graphs are perfect, but a thick bipartite graph is not necessarily perfect. See Fig. 4. More obviously, a tree has no holes, but a link may have an exponential number of 4-holes. By contrast, $\text{full}(\mathcal{C})$ inherits many properties from $\mathcal{C}$.

The following is essentially the content of section 4.2.1 of [19].

Lemma 2 (Dyer, Greenhill, Müller). *Let $\mathcal{F}$ be a set of graphs, none of which contains true twins, and let $\mathcal{C}$ be the graph class with minimal forbidden subgraphs $\mathcal{F}$. Then $\text{full}(\mathcal{C}) = \mathcal{C}$.*

Proof. Clearly $H \in \text{full}(H)$, because a vertex is a 1-clique and an edge is a 2-clique. Thus $\mathcal{C} \subseteq \text{full}(\mathcal{C})$. That $\text{full}(H) \subseteq \mathcal{C}$ if $H \in \mathcal{C}$ is Lemma 5 of [19]. $\square$

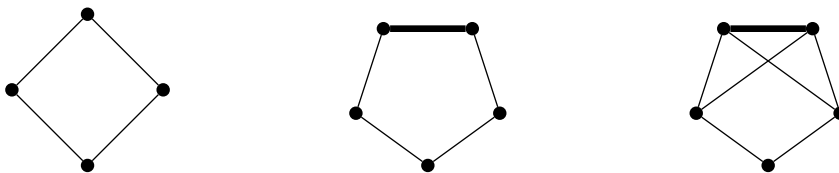
Thus, for classes $\mathcal{C}$ satisfying the assumption of Lemma 2, $\text{full}(\cdot)$ is the identity operator. Many hereditary classes satisfy Lemma 2: unit interval, chordal, weakly chordal, perfect and even-hole-free graphs, for example.

It is also proved in Lemma 5 of [19] that, for any class having a forbidden subgraph containing true twins, Lemma 2 no longer holds. Thus it is false, for example, in any subclass of triangle-free graphs, since any two vertices of a k -clique are true twins for all $k > 2$.

Lemma 2 can be useful. It was used in [19] to simulate vertex weights in the independent set problem for graphs of bounded bipartite pathwidth. It was used by Lovász [36] in the proof of the (weak) PGT, although without explicitly assuming that the class $\mathcal{P}$ satisfies the condition of Lemma 2, which follows from the SPGT.

The recognition problem for $\text{thick}(\mathcal{C})$ is the same as that for $\mathcal{C}$ for any class satisfying Lemma 2, but that is not true for subclasses. For example, $\mathcal{B} \subset \mathcal{P}$, and $\mathcal{P}$ satisfies Lemma 2, so $\text{thick}(\mathcal{P}) = \mathcal{P}$. But, even though the recognition problem for $\mathcal{B}$ is almost trivial, and recognition algorithms for $\mathcal{P}$ are known [12], the status of the recognition problem for $\text{full}(\mathcal{B})$ is unclear. Even recognising $\text{full}(\mathcal{F})$ is a different question from recognising $\text{thick}(\mathcal{F})$, even though $\text{full}(\mathcal{F}) \subset \text{thick}(\mathcal{F})$. For example, a k -star has $k+1$ models in $\text{thick}(\mathcal{F})$, but a unique model in $\text{full}(\mathcal{F})$. On the other hand, graphs in $\text{full}(\mathcal{F})$ are chordal, whereas those in $\text{thick}(\mathcal{F})$ are generally not.

So, full graph classes are not without interest, but we do not consider them further here.

Fig. 4. C_4 , a thick C_4 and a full C_4 .

2.2. Subcolouring

The *subchromatic number* [2] $\chi_s(G)$ of G is $\min\{\chi(H) : H \in \text{thin}(G)\}$ is. A graph is *q-subcolourable* if $\chi_s(G) \leq q$. This is equivalent to asking if $G \in \text{thick}(\mathcal{C}_q)$, where $\mathcal{C}_q$ is the class of q -colourable (q -partite) graphs.

A graph G has $\chi_s(G) = 1$ if and only if it is a *cluster graph*, or $G \in \text{thick}(\mathcal{I})$, where $\mathcal{I}$ is the class of independent sets. Also $\chi_s(G) = 2$ if and only if $G \in \mathcal{B}$, where $\mathcal{B} = \mathcal{C}_2$ is the class of bipartite graphs. We will be interested in the class $\text{thick}(\mathcal{F})$ of *thick forests*, where $\mathcal{F} \subset \mathcal{B}$ is the class of *forests*. We consider the recognition problem for thick forests in Section 4, and counting independent sets and colourings in sections 6.1 and 6.2.

A *generalised q-colouring* of a graph $G = (V, E)$ with a graph class $\mathcal{C}$ is a partition of V into subsets $V_1, V_2, \dots, V_q$ such that $G[V_i] \in \mathcal{C}$ for all $i \in [q]$. If $\mathcal{C}$ is the class of cluster graphs, then a generalised q -colouring is a *q-subcolouring* [2]. So G has 1-subcolouring if it is a cluster graph and has a 2-subcolouring if it is a thick bipartite graph. Unfortunately, whether G has a q -subcolouring is an NP-complete question for any $q \geq 2$ [23] (see Fig. 5).

Lemma 3. *It is NP-complete to decide whether $G \in \text{thick}(\mathcal{C}_q)$ for any $2 \leq q = \mathcal{O}(n^{1-\varepsilon})$, where $0 < \varepsilon < 1$ is any constant.*

Proof. This follows from a result of Farrugia [23] (see also [10]) on generalised colourings. Let $\mathcal{C}_1, \mathcal{C}_2$ be two additive hereditary graph classes with recognition in NP, neither being the class of edgeless graphs. Let ψ be a 2-colouring of an input graph $G = (V, E)$ with $C_i = \psi^{-1}(i)$ so that $G[C_i] \in \mathcal{C}_i$ for $i \in [2]$. Then it is NP-complete to decide whether such a 2-colouring exists. For a bipartite thick graph, we take $\mathcal{C}_1$ and $\mathcal{C}_2$ to be the cluster graphs. This class is clearly additive and hereditary, and recognition is in P. Then the existence of ψ is equivalent to asking whether G is a thick bipartite graph.

This can then be extended to asking whether G is a thick q -partite graph by inductively taking the same $\mathcal{C}_1$ and taking $\mathcal{C}_2$ to be $\text{thick}(\mathcal{C}_{q-1})$. This remains valid if $q = \mathcal{O}(n^{1-\varepsilon})$, since polynomial in n^ε is equivalent to polynomial in n . $\square$

Note that the additive condition used in Lemma 3 is important, or recognition of unipolar graphs would be NP-complete when it is not, as we discuss in Section 4.2. Also, Lemma 3 does not necessarily hold if G is restricted to lie in some graph class. For example, deciding whether $G \in \text{thick}(\mathcal{C}_q | \mathcal{C})$ is shown in [9] to be in P if $\mathcal{C}$ is the class of interval graphs or permutation graphs. We note that these three classes are subclasses of perfect graphs $\mathcal{P}$. However perfection is clearly not a sufficient condition, since $\mathcal{B} \subset \mathcal{P}$. Moreover, q -subcolouring for $q > 2$ is NP-complete even in the class of chordal graphs [41]. That is, the recognition problem for $\text{thick}(\mathcal{C}_q | \mathcal{C}_\Delta)$ is NP-complete for $q > 2$.

2.3. Altering the model

Given a model (H, ψ) of G we will consider how ψ can be changed so that H' corresponds to an operation on H . In particular, we are interested in modifications which give $H' = H$.

Given G and $H, H' \in \text{thin}(G)$ we say that $u \in \mathbf{u} \in \mathcal{V}$ is *movable* to $\mathbf{w} \in \mathcal{V}'$ if the model (H, ψ) of G can be modified to a model (H', ψ') of G so that $\psi'(u) = \mathbf{w}' = \mathbf{w} \cup \mathbf{u}$ and $\psi'(v) = \psi(v)$ for all $v \neq u$.

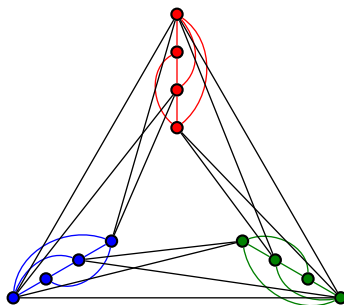


Fig. 5. A thick triangle with a 3-subcolouring.

Lemma 4. A vertex $u \in \mathbf{u}$ is movable to $\mathbf{w}$ if and only if $\mathbf{u} \cup \mathbf{w} \subseteq N[u]$.

Proof. We must have $\mathbf{w} \cup u$ a clique, which is true if and only if u is complete to $\mathbf{w}$, which is equivalent to $\mathbf{w} \subseteq N(u)$. Also $\mathbf{u} \setminus u \subseteq N(u)$ since $\mathbf{u}$ is a clique. Conversely, if $\mathbf{w} \cup \mathbf{u} \not\subseteq N[u]$ then there is a $w \in \mathbf{w}$ such that $uw \notin E(G)$. So $\mathbf{w} \cup u$ is not a clique, a contradiction. $\square$

Corollary 1. A vertex $u \in \mathbf{u} \in \mathcal{V}$ is movable to $\mathbf{w} \in \mathcal{V}'$ only if $\mathbf{u}\mathbf{w} \in \mathcal{E}$ or $\mathbf{w} = \{u\}$. $\square$

It follows that $\text{thin}(G)$ is connected by vertex moves, since any $H \in \text{thin}(G)$ is connected to $G \in \text{thin}(G)$ by a sequence of $\mathcal{O}(n)$ moves, by dismantling the nodes of H to single vertices, then reassembling them to give H' .

Most moves in G change H , since moving u can produce a new edge in H' . We will call move the *good* if $H' = H$ after the move. We could give a general criterion for move to be good, but we will restrict attention to the triangle-free case, $H \in \mathcal{T}$.

Lemma 5. If $H \in \text{thin}(G) \cap \mathcal{T}$ and $\mathbf{u}, \mathbf{w} \in H$ then moving $u \in \mathbf{u}$ to $\mathbf{w}$ is a good move if and only if $\{u\} \subset \mathbf{u}$ and $N[u] = \mathbf{u} \cup \mathbf{w}$. The reverse of a good move is a good move.

Proof. If $\{u\} \subset \mathbf{u}$ and $N[u] = \mathbf{u} \cup \mathbf{w}$ then moving u to $\mathbf{w}$ does not change H , since u is movable by Lemma 4 and has no neighbours outside $\mathbf{u}$ and $\mathbf{w}$. Conversely, suppose $v \in N(u)$ but $v \notin \mathbf{u} \cup \mathbf{w}$. Then $u \in \mathbf{w}' \neq \mathbf{w}$ for some neighbour $\mathbf{w}'$ of $\mathbf{u}$. Thus moving u produces an edge uv between $\mathbf{w} \cup u$ and $\mathbf{w}'$, so the resulting $H' \in \text{thin}(G)$ contains a triangle $\mathbf{u} \setminus u, \mathbf{w} \cup u, \mathbf{w}'$, so $H' \notin \mathcal{T}$ and hence $H' \neq H$. If $\{u\} = \mathbf{u}$, then moving u to $\mathbf{w}$ contracts the edge $\mathbf{u}\mathbf{w}$ in H to the vertex $\mathbf{w}$ in H' and so $H' \neq H$. Good moves are reversible from the symmetry of $N[u] = \mathbf{u} \cup \mathbf{w}$. $\square$

In fact, we will also call a move good if $\mathbf{w} = \emptyset$. In this case we have $N[u] = \mathbf{u}$, and the move corresponds to H having a new leaf vertex $\mathbf{w}' = \{u\}$. The reverse move then contracts the full link $\mathbf{w}'\mathbf{u}$ to $\mathbf{u}$.

Successive moves are even more restricted if we require $H' = H \in \mathcal{T}$.

Lemma 6. If $H \in \text{thin}(G) \cap \mathcal{T}$ and $u_1, u_2 \in \mathbf{u}$ have good moves to $\mathbf{w}_1, \mathbf{w}_2$ respectively. Then both can be moved so that $H' = H \in \mathcal{T}$ if and only if $\mathbf{w}_1 = \mathbf{w}_2$ and $\{u_1, u_2\} \subset \mathbf{u}$.

Proof. Both $\mathbf{w}_1, \mathbf{w}_2$ are neighbours of $\mathbf{u}$. If $\mathbf{w}_1 \neq \mathbf{w}_2$, moving gives $\mathbf{u}' = \mathbf{u} \setminus \{u_1\}$ and $\mathbf{w}'_1 = \mathbf{w}_1 \cup \{u_1\}$. But moving u_2 to $\mathbf{w}_2$ is a good move only if $\mathbf{h}' \cup \mathbf{w}_2 = N(u_2)$, which is false since $u_1 \in N(u_2)$, but $u_1 \in \mathbf{h}' \cup \mathbf{w}_2$ only if $\mathbf{w}_1 = \mathbf{w}_2$.

If $\mathbf{w}_1 = \mathbf{w}_2$, both can be moved in either order unless $\{u_1, u_2\} = \mathbf{u}$, in which case moving both replaces the path $\mathbf{w}_1\mathbf{u}\mathbf{w}_2$ by an edge $\mathbf{w}_1\mathbf{w}_2 \notin \mathcal{E}$, since $H \in \mathcal{T}$. Thus $H' \neq H$. $\square$

Corollary 2. If $H \in \text{thin}(G) \cap \mathcal{T}$, a subset $U \subseteq \mathbf{u}$ can have good moves to at most one of $\mathbf{u}$'s neighbours $\mathbf{w}$ so that $H' = H$. $\square$

While all models of G are connected by moves, they need not be connected by good moves. We will call a model *rigid* if it has no good moves. Consider G the 4-cycle $(abcd)$ with H a single edge $\mathbf{u}\mathbf{w}$. There are two models $\mathbf{u} = \{a, b\}$, $\mathbf{w} = \{c, d\}$ or $\mathbf{u} = \{a, d\}$, $\mathbf{w} = \{b, c\}$, but both are rigid. Thus neither is connected to the model (G, id) by good moves.

Contracting a full link $\mathbf{u}\mathbf{w}$ in H corresponds to regarding it as a node $\mathbf{u} \cup \mathbf{w}$ in H' , that is, contracting the edge of H . This may also be viewed as moving all vertices from $\mathbf{u}$ into $\mathbf{w}$, or vice versa. Note that these are not necessarily good moves. We used this operation in Section 2.1 to eliminate loose triangles and 4-cycles. But, if $H \in \mathcal{T}$, and we require $H' \in \mathcal{T}$, we cannot contract an edge of a stiff 4-cycle, or we obtain a stiff triangle.

We will call the reverse operation *splitting* the thick vertex $\mathbf{u}$. We divide $\mathbf{u}$ into disjoint subsets $\mathbf{u}_1, \mathbf{u}_2$ so that $\mathbf{u}_1\mathbf{u}_2$ becomes a full link in H' . If $H \in \mathcal{T}$ and we require $H' \in \mathcal{T}$, we must ensure that $N(\mathbf{u}_1)$ and $N(\mathbf{u}_2)$ are disjoint. Also, we cannot usefully split $\mathbf{u}_1$ or $\mathbf{u}_2$, or we introduce a loose triangle.

We have seen that cannot hope to recognise the whole class $\text{thick}(\mathcal{T})$, since $\mathcal{B} \subset \mathcal{T}$ and we know from Lemma 3 that recognition of $\text{thick}(\mathcal{B})$ is NP-complete. So our approach to recognition of graphs in $\text{thick}(\mathcal{T})$ will be based on another property of triangle-free graphs, that the induced neighbourhood every vertex is a star. This corresponds to a *unipolar* subgraph of G . Recall the definition from Section 1.1. We use this to try to construct H one vertex at a time and then, if necessary, move vertices as described above to ensure consistency.

So we require recognition of unipolar graphs, which is in polynomial time. See [22,38,43] and Section 4.2. But the following underlies our approach.

Lemma 7. The model (H, ψ) of a unipolar graph G is

- (a) unique up to good moves between the hub and one of its satellites;
- (b) rigid, but with a possible choice between two models.

Proof. We assume a hub $\mathbf{h}$ with satellites $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_s$, and model (H, ψ) . If $\mathbf{h}$ is given, then $G \setminus \mathbf{h}$ is a cluster graph, so $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_s$ are uniquely identified. If $u \in \mathbf{b}_i$, we cannot move to $\mathbf{b}_j$ for $i \neq j$ by Lemma 4, since $\mathbf{b}_j \cap N(u) = \emptyset$. We can move $u \in \mathbf{b}_i$ to $\mathbf{h}$ by Lemma 4 if $\mathbf{b}_i \cup \mathbf{h} \subseteq N(u)$. But clearly $N(u) \subseteq \mathbf{b}_i \cup \mathbf{h}$, so we must have $\mathbf{b}_i \cup \mathbf{h} = N(u)$. Thus this is a

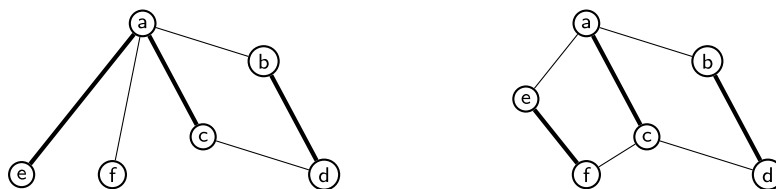


Fig. 6. Case (b) of Lemma 7.

good move by Lemma 5. We cannot move vertices into more than one satellite by Cor. 2. This establishes the “normal” case (a), which is sufficient if G is chordal. See [22].

However, there is another case, where h has a subclique a such that $G \setminus a$ has a thick P_3 component $\{b, c, d\}$. See Fig. 6. Then $(abcd)$ is a thick 4-cycle. This is a thick edge if either of the pairs of links ab, cd or ac, bd are both full. If as shown in Fig. 6, ac, bd are full links, then ac is the hub, bd is a satellite and G is unipolar. If all of ab, cd, ac, bd are full, and $(abcd)$ is the only thick 4-cycle, then there is a choice of whether ab or ac becomes the hub.

However, there can be more than one such 4-cycle. Each must contribute a full link to the hub so as to be contained in a thick edge. Thus, if there is more than one, the hub must be their unique common full link for G to be unipolar. An example is shown in Fig. 6, where ac is the hub and bd, ef are the satellites.

Note also that the model is rigid. There are no good moves from any satellite because no vertex in any satellite is complete to the hub (ac in Fig. 6). Since good moves are reversible, neither can there be any from the hub into the satellites. $\square$

3. Thick forests

A thick tree G with 26 vertices and 49 edges is shown in Fig. 7. If we take the nodes to be the horizontal K_2 's, the thin tree has 4 internal vertices, 9 leaves and 12 edges. In fact, it is easy to see that G is a chordal graph, so it is a chordal thick tree.

We have shown in Section 2.2 that even recognising a thick bipartite graph is hard in general. By contrast, we will show that the class of thick forests $\text{thick}(\mathcal{F}) \subset \text{thick}(\mathcal{B})$ is tractable for most purposes. However, some of its properties of $\text{thick}(\mathcal{F})$ are inherited from $\text{thick}(\mathcal{B})$, so we first consider this class.

3.1. Thick bipartite graphs

We have the following characterisation.

Lemma 8. $G \in \text{thick}(\mathcal{B})$ if and only if it has a vertex 2-colouring such that each colour class has no induced P_3 .

Proof. P_3 is the only forbidden subgraph for a cluster graph, and $G \in \text{thick}(\mathcal{B})$ if and only if its thin graph is 2-colourable. $\square$

From this we have the following.

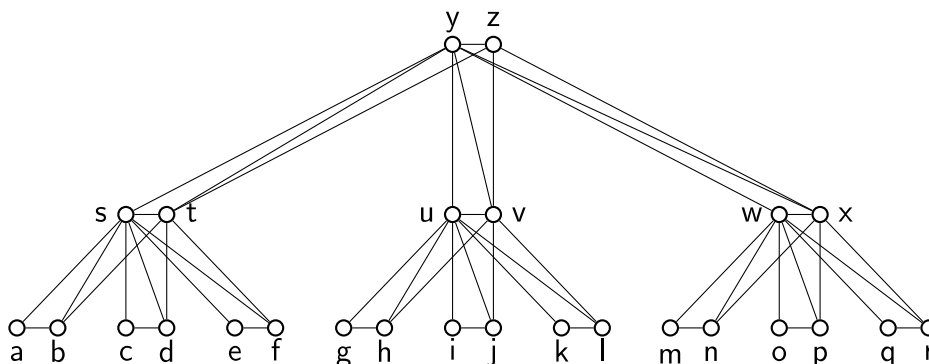


Fig. 7. A chordal thick tree.

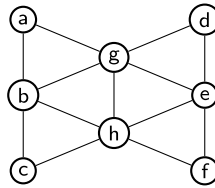


Fig. 8. A chordal graph not in $\text{thick}(\mathcal{F})$.

Lemma 9. A thick bipartite graph has no odd antihole of size greater than five.

Proof. Consider a 2-colouring with colours red and blue of an antihole $\bar{C}_k$ where $k \geq 7$ and odd. Suppose the vertices of C_k are $v_0, v_1, \dots, v_{k-1}$ cyclically, with subscripts mod k . Since k is odd, there must be two adjacent vertices in C_k , v_1, v_2 say, which have the same colour, say red. These are nonadjacent in $\bar{C}_k$, but have all of $v_4, v_5, \dots, v_{k-1}$ as common neighbours. These $(k-4)$ vertices must therefore all be coloured blue, or we would have a red P_3 . Now v_0 and v_3 cannot both be red, or there would be a red P_3 : v_0, v_3, v_1 . So there can be at most three red vertices and, by symmetry, at most three blue vertices. This is a contradiction, since $k \geq 7$. $\square$

From this we have

Lemma 10. A thick bipartite graph G is perfect if and only if it has no odd holes.

Proof. From the SPGT, we need only show that G contains no odd antihole. But $\bar{C}_5 = C_5$ excludes 5-antiholes, and Lemma 9 excludes larger odd antiholes. $\square$

Let $\mathcal{P}_0$ denote the class of graphs without long holes or odd antiholes, as defined in [40]. Clearly $\mathcal{P}_0 \subset \mathcal{P}$, from the SPGT. So $\mathcal{P}_0$ is the class of perfect graphs without long holes. Thus we have

Corollary 3 (Salamon and Jeavons [40]). $\text{thick}(\mathcal{F}) \subseteq \mathcal{P}_0$.

Proof. If $G \in \mathcal{F}$, a chordless cycle must be contained in a thick edge. Thus G can only have triangles and 4-holes. The result then follows from Lemma 10. $\square$

For thick graph classes, this has a converse.

Lemma 11. Let $\mathcal{C}$ be a class of graphs. Then $\text{thick}(\mathcal{C})$ is a class of perfect graphs if and only if $\mathcal{C} \subseteq \mathcal{F}$.

Proof. If $\mathcal{C} \subseteq \mathcal{F}$, this is Corollary 3. Conversely, suppose there exists $H \in \mathcal{C} \setminus \mathcal{F}$. Then H has a triangle or hole, C . If C is an odd hole, $H \notin \mathcal{P}$ by the SPGT, so $\text{thick}(H) \not\subseteq \mathcal{P}$. If C is an even hole, then we replace any vertex $v \in C$ with a K_2 vv' so that an edge vw of C becomes $vv'w$, to give a graph H' with odd hole C' . (See Fig. 4.) Then $H' \in \text{thick}(H)$, but $H' \notin \mathcal{P}$, so $\text{thick}(H) \not\subseteq \mathcal{P}$. If C is a triangle uvw , we replace it similarly with K_2 's uu', vv' so that uvw becomes a 5-hole $uu'vv'w$ in a graph $H' \in \text{thick}(H)$ such that $H' \notin \mathcal{P}$. $\square$

Thus $\mathcal{F}$ is the largest class $\mathcal{C}$ for which $\text{thick}(\mathcal{C})$ contains only perfect graphs.

However, not all perfect graphs without long holes are thick forests. See, for example Figs. 8 and 9. We defer proof of the claims to Section 3.2.

The complexity of recognising $G \in \text{thick}(\mathcal{B}|\mathcal{P}_0)$ is thus an interesting question. This is not obviously in P, even for subclasses of $\mathcal{P}_0$, for example the class $\mathcal{C}_\Delta$ of chordal graphs. Stacho [41] gave a polynomial time algorithm for recognition of $G \in \text{thick}(\mathcal{B}|\mathcal{C}_\Delta)$. However, we show below, in Corollary 4, that $\text{thick}(\mathcal{B}|\mathcal{C}_\Delta) = \text{thick}(\mathcal{F}|\mathcal{C}_\Delta)$. So recognition for $\text{thick}(\mathcal{B}|\mathcal{C}_\Delta)$ is implied by recognition for $\text{thick}(\mathcal{F})$, which we show in Section 4.

Eschen, Hoàng, Petrick and Sritharan [21] defined a subclass of $\mathcal{P}_0$ which they called *biclique separable* graphs. This is the hereditary class of graphs which are either cliques, or have a separator consisting of two disjoint cliques, and are also

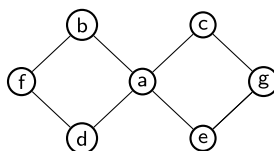


Fig. 9. A bipartite graph not in $\text{thick}(\mathcal{F})$.

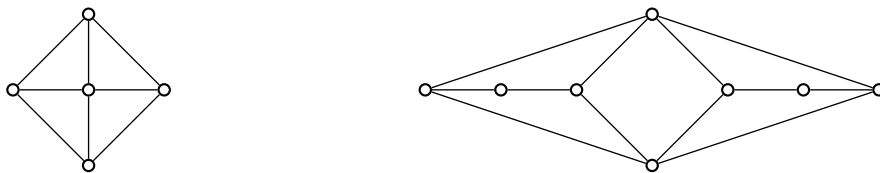


Fig. 10. Biclique separable is incomparable with $\mathcal{F}$.

long hole-free. This class strictly includes *weakly chordal* graphs, but does not include $\text{thick}(\mathcal{F})$. The graph on the left in Fig. 10 is an example. This graph is cobipartite, so in $\text{thick}(\mathcal{F})$, is not a clique and has no clique separator. The only nonadjacent cliques are the two pairs of opposite corner vertices. Removal of either of these pairs does not disconnect the graph, so it is not biclique separable. Neither is every biclique separable graph in $\mathcal{F}$. The graph on the right in Fig. 10 is an example. It is biclique separable, in fact weakly chordal, but is not a clique, not cobipartite and has no clique separator. So it is not in $\mathcal{F}$.

The class $\mathcal{P}_0$ has not received much attention. Recognition is clearly in P, using the algorithms of [12,13] on $\bar{G}$. But we can ask algorithmic questions about $\mathcal{P}_0$ which might be easier to answer than for $\mathcal{P}$. For example, is there a combinatorial algorithm for finding a maximum independent set in $G \in \mathcal{P}_0$? More ambitiously, can we count independent sets in polynomial time for $G \in \mathcal{P}_0$? We show below that this can be done for $G \in \mathcal{F}$, but is #P-complete for $G \in \mathcal{P}$.

3.2. Forbidden subgraphs

Since they are a hereditary class, thick forests can be characterised by minimal forbidden subgraphs, but there appear to be too many to be useful. Some examples are given in Fig. 11, but we will not prove these, except for the graph of Fig. 11. The others can be proved similarly. There are even infinite families of minimal forbidden subgraphs besides long holes and odd antiholes. One such family is shown in Fig. 12. We will not prove this, but it can be shown not to be a thick tree using the algorithm of Section 4, and minimal likewise, using the symmetries. We conjecture that all such infinite families of forbidden subgraphs for $\mathcal{F}$ are of a similar form, two small graphs connected by a chain of triangles.

However, a simpler criterion can be given using forbidden *coloured* subgraphs:

Theorem 1. A graph G is a thick forest if and only if it has a vertex 2-colouring which excludes

- (a) a long hole (with any colouring),
- (b) a monochromatic P_3 ,
- (c) a 4-hole with the alternating colouring. (See Fig. 13.)

Proof. Suppose G is a thick forest and $H \in \text{thin}(G)$. Then G is a thick bipartite graph, so must have a 2-colouring satisfying (b) by Lemma 8. It must satisfy (a) by Lemma 14. Also, by Proposition 2, any 4-hole must lie in a thick edge, so it must have the cobipartite colouring (see Fig. 13), and cannot have the alternating colouring. Hence (c) is satisfied.

Conversely, if G has a 2-colouring satisfying (b), that is, a 2-subcolouring, it is a thick bipartite graph. Also H cannot have a triangle or odd hole. If H has a k -hole, G has hole of size $k \leq \ell \leq 2k$, by alternately using an edge of G in a link and an edge of G in a node, if necessary. Thus any hole must have $k \leq 4$, or (a) would be contradicted. Thus we have only to consider the case where G has a 4-hole. The only 2-colouring of a 4-hole not ruled out by (b) and (c) is the cobipartite colouring. But a 4-hole of G with this colouring must lie in a link since its monochromatic edges must lie in two different nodes and its dichromatic edges must lie in the edge set connecting them. Thus H can have no triangles or holes and must be a forest. $\square$

Corollary 4. A chordal graph is 2-subcolourable if and only if it is a thick forest.

Proof. If G is chordal conditions (a) and (c) of Theorem 1 are automatically satisfied, since then there are no 4-holes or longer in G . Condition (b) is equivalent to a G having a 2-subcolouring. $\square$

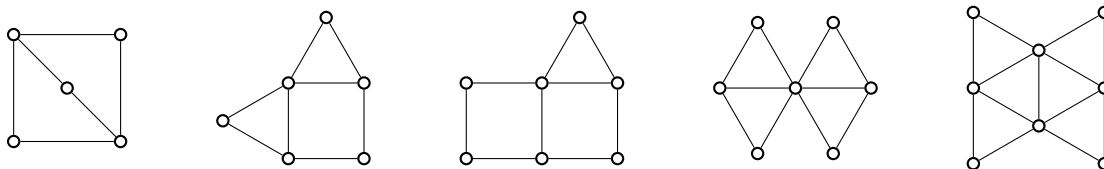


Fig. 11. Some minimal forbidden subgraphs for thick forests.

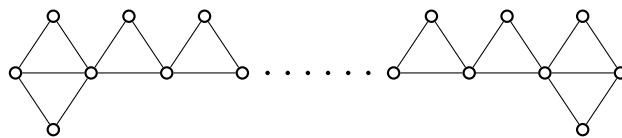


Fig. 12. An infinite family of minimal forbidden subgraphs for thick forests.

Thus the graph class considered in [41] is precisely the class of chordal thick forests. In our notation, we have shown that $\text{thick}(\mathcal{B}|\mathcal{C}_\Delta) = \text{thick}(\mathcal{F}|\mathcal{C}_\Delta)$.

Note that, if G is chordal, its links must be a cobipartite chordal graphs. These are cochain graphs, since the complement is bipartite and has no $\tilde{C}_4 = 2K_2$ so is a chain graph.

This forbidden coloured subgraph characterisations does not imply a polynomial time recognition algorithm. Recognising a thick bipartite graph, that is, a graph satisfying only (b), is NP-complete from Lemma 3. On the other hand, monochromatic forbidden subgraph characterisations do not imply polynomial time recognition when there are infinitely many forbidden subgraphs, as there are here. See, for example, [30].

Theorem 1 casts the recognition of thick forests as a instance of SAT if the vertices are variables and the two colours represent true and false. Then the clauses derived from (b) are of type NAE (not-all-equal) 3SAT, and those derived from (c) are of type NAE (not-all-equal) 2SAT. Thus the problem is an instance of NAE 2/3SAT. In general these problems have NP-complete decision. So this does not lead to a decision algorithm, and our approach in Section 4 does not use this characterisation of thick forests.

However, we will apply Theorem 1 to show that the graphs of Figs. 8 and 9 are not thick trees.

Consider the graph G of Fig. 8. Suppose first that g and h have the same colour, without loss of generality red. Then a, c, d, f must all be blue, or one of the P_3 's hga, hgd, ghc, ghf would be monocoloured red. But now b, e must both be red, or one of the P_3 's abc, def would be monocoloured blue. But now we have P_3 's bge, bhe monocoloured red, so this is a bad colouring. So g and h have different colours, g red and h blue without loss of generality. Now either a or d must be blue. Otherwise we have a red P_3 agd . Assume, by symmetry, that a is blue. Now b must be red or we have a blue P_3 abh . Then c, e must be blue or we have a red P_3 gbc or bge . But now there is a blue P_3 che so again we have a bad colouring. Therefore, by Theorem 1, G is not a thick tree.

The graph G of Fig. 9 has two C_4 's. Thus b, d and c, e must receive different colours. Assume, by symmetry, that b, c are red and d, e are blue. Then we cannot colour a red or we would have a red P_3 bac and we cannot colour a blue or we would have a blue P_3 dae . Therefore, by Theorem 1, G is not a thick tree.

In fact, both the graphs of Figs. 8 and 9 are minimal forbidden subgraphs for the class of thick forests, since it is easy to see that deleting any vertex leaves a thick forest.

4. Recognising a thick forest

We have seen in Lemma 3 that it is NP-complete to recognise the class $\text{thick}(\mathcal{B})$ of thick bipartite graphs. However, this intractability does not necessarily apply to subclasses of $\text{thick}(\mathcal{B})$. Thus we will give an algorithm to recognise a thick forest in polynomial time. If the recognition algorithm succeeds, we will say that G is *accepted*, and we terminate with an explicit $H \in \text{thin}(G) \cap \mathcal{F}$. Otherwise, $\text{thin}(G) \cap \mathcal{F} = \emptyset$, and we will say that G is *rejected*.

Remark 1. Note that we do not guarantee to recover any particular representation of G as a thick tree. In fact, $\text{thin}(G)$ is usually exponentially large, as discussed in Section 2. Rather, we construct some $H \in \text{thin}(G) \cap \mathcal{F}$, if one exists, with failure meaning that $\text{thin}(G) \cap \mathcal{F} = \emptyset$. Thus, in what follows, nodes $u, w, \dots$ should not be thought of as fixed in advance, but rather as vertices of some $H \in \text{thin}(G) \cap \mathcal{F}$.

Stacho [41] gave a polynomial time algorithm to decide whether a chordal graph is 2-subcolourable. From Corollary 4, we know that this is the same problem as recognising that a chordal graph as a thick forest. Thus, for example, Stacho's algorithm should verify that the graph of Fig. 7 is a thick tree. However, the algorithm is restricted to chordal thick trees, and does not appear to explicitly yield a thick tree representation. So we give a different algorithm here, which recognises a general thick forest with the same time complexity.



Fig. 13. Alternating and cobipartite colourings of a 4-hole.

To decide if $G = (V, E)$ is a thick forest, it suffices to decide whether each component of G is a thick tree, so we will consider only the case where G is a thick tree. Then our algorithm is based on the simple fact that the structure of a tree is precisely that it can be decomposed into disjoint trees by the removal of any vertex. Thus, if we can identify any node $\mathbf{u}$ of G , we can cast the recognition problem as recursively recognising the subtrees rooted at its children $\mathbf{w}_i$ ($i \in [r]$). We terminate the recursion when the subtree is a clique, a thick leaf of $\text{thin}(G)$.

For any node $\mathbf{u}$ of a thick forest, its induced neighbourhood $G[N(\mathbf{u})]$ must be a cluster graph, and hence $G[N(\mathbf{u})]$ is a unipolar graph with $\mathbf{u}$ as its hub and satellites comprising subcliques C_i of its thick neighbours $\mathbf{w}_i$ ($i \in [r]$), where $r = \deg \mathbf{u}$ in $H \in \text{thin}(G)$.

To identify a $\mathbf{u} \in \text{thin}(G)$, use the following. Suppose $G \in \text{thick}(\mathcal{T})$ and we have a clique C such that $C \subseteq \mathbf{u}$ for some node $\mathbf{u}$. Then $N[C]$ induces a unipolar graph G_C with hub $\mathbf{h} \subseteq \mathbf{u}$.

We use the algorithm UNIPOLAR in Section 4.2 to decompose G_C . A necessary ingredient is recognition of a link, a cobipartite graph. We show this in Section 4.1.

While $\mathbf{h} = \mathbf{u}$ is the “usual” case in the unipolar decomposition, we must also deal with cases where $\mathbf{h} \subset \mathbf{u}$. We describe our method for this in Section 4.3.

To initialise the algorithm we choose an arbitrary $v \in V$, which must be in some $\mathbf{u}$, and take $C = \{v\} \subseteq \mathbf{u}$. We then apply this recursively to the subtrees rooted at $\mathbf{u}$'s neighbours $\mathbf{w}_i$ ($i \in [r]$), using the clique $C_i \subseteq \mathbf{w}_i$ from the cluster graph $G[N(\mathbf{u})]$ as seed to determine the $\mathbf{w}_i$. The recursion terminates when a subtree is simply a node. The totality of nodes that appear in the algorithm are then those of the tree $\text{thin}(G)$, and we accept G . If any failure occurs in the algorithm, we stop, concluding that G is not a thick tree and reject it.

This essentially completes the description of the algorithm. However, there is a remaining difficulty, due to subgraphs of G which can be viewed as links in more than one way. In the above algorithm, this can result in a false rejection of G . We consider how to recognise and resolve this problem in Section 4.4.

Finally, in Section 4.5, we show that the time complexity of the algorithm is $\mathcal{O}(mn)$.

4.1. Recognising a cobipartite graph

Lemma 12. A cobipartite graph $G = (V_1 \cup V_2, E)$, with possible pre-assignments of vertices to V_1 and V_2 , is recognisable in $\Theta(m)$ time.

Proof. The following simple algorithm recognises a cobipartite graph by recognising its complement as a bipartite graph.

COBIPARTITE

- (1) Form the complementary graph $\bar{G} = (V_1 \cup V_2, \bar{E})$ in $\mathcal{O}(n^2)$ time.
- (2) Determine the connected components $B_1, \dots, B_k$ of $\bar{G}$ in $\mathcal{O}(n^2)$ time.
- (3) Check whether any B_i is not bipartite in $\mathcal{O}(n^2)$ time. If so, reject G . Otherwise, let $B_i = B_{i1} \cup B_{i2}$ ($i \in [k]$) be the bipartitions.
- (4) Determine V_1 and V_2 by assigning B_{i1}, B_{i2} to V_1 and V_2 for $i \in [k]$, taking into account pre-assignments. If these conflict, reject G . Otherwise put the larger of B_{i1}, B_{i2} in V_2 or choose arbitrarily if $|B_{i1}| = |B_{i2}|$.

Maximising the size of V_2 in step (4) is a choice we make for use in UNIPOLAR below. In general, ignoring pre-assignments, there are 2^k ways of assigning the bipartitions.

To complete the proof we need only show that $n^2 = \Theta(m)$. Let $n_1 = |V_1|$, $n_2 = |V_2|$, so $n = n_1 + n_2$ and $|V_1 : V_2| \leq n_1 n_2$. Then

$$\frac{1}{4}n(n-1) \leq \frac{1}{2}n_1(n_1-1) + \frac{1}{2}n_2(n_2-1) \leq m \leq \frac{1}{2}n_1^2 + \frac{1}{2}n_2^2 + n_1n_2 \leq n_1^2 + n_2^2 \leq n^2,$$

where we have used the convexity of $x(x-1)$ with $x = n/2$ in the first inequality, and $2n_1n_2 \leq n_1^2 + n_2^2$ in the fourth. Thus $m = \Theta(n_1^2 + n_2^2) = \Theta(n^2)$, and so any algorithm recognising G must be $\Theta(m) = \Theta(n^2)$. $\square$

Note that we may be able to improve the time bound if there are sufficiently many pre-assignments, but we will not pursue this here.

4.2. Recognising a unipolar graph

We first show how we can recognise a unipolar graph, a thick star, with hub $\mathbf{h}$ and satellites $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_s$. From Lemma 7 we know that a unipolar decomposition is not necessarily unique, so we make the following assumption:

(*) As many vertices as possible are moved from the hub into one satellite.

Let $\mathbf{b}_{s+1} \leftarrow \emptyset$. Then, if $N(v) = \mathbf{h} \cup \mathbf{b}_i$ for any $i \in [s+1]$, we can move v from $\mathbf{h}$ into $\mathbf{b}_i$ by Lemma 5. Moving all such v from $\mathbf{h}$ into $\mathbf{b}_i$ will ensure that (*) is satisfied at the end of the algorithm. We can only choose one such i by Corollary 1. If now $\mathbf{b}_{s+1} \neq \emptyset$, set $s \leftarrow s+1$, otherwise delete $\mathbf{b}_{s+1}$. The purpose of (*) is that $\mathbf{h}$ is minimal. This is useful in Section 4.3, since it means that every vertex in $\mathbf{h}$ must be in the hub.

Unipolarity can be recognised by existing methods [22,38,43], but we will now give a simple $\mathcal{O}(mn)$ time algorithm using the same ideas as for a general thick tree. The approach is closest to that in [43] in that it uses only vertex

neighbourhoods and testing for cobipartiteness, which we can check using the algorithm COBIPARTITE given in Section 4.1.

The justification is given in the bullet point comments.

UNIPOLAR

- (1) $i \leftarrow 1, V_1 \leftarrow V$.
- (2) $G_i \leftarrow G[V_i]$. Choose $u_i, w_i \in V_i$ such that $u_i w_i$ is a nonedge of G_i . If no u_i, w_i exist, $\mathbf{h} \leftarrow G_i$ and stop, we have found a unipolar decomposition.
 - G_i is a clique, so must be $\mathbf{h}$. Otherwise, the nonedge $u_i w_i$ is used in (6).
- (3) Let $U_i \leftarrow N[u_i] \cap V_i, \bar{U}_i \leftarrow V \setminus U_i$.
 - Loop initialisation. The node $\mathbf{u}_i$ of G containing u_i (either $\mathbf{h}$ or $\mathbf{b}_i$) satisfies $\mathbf{u}_i \subseteq U_i$. If $\mathbf{u}_i = \mathbf{b}_i$, U_i is an initial estimate of $\mathbf{b}_i$.
- (4) Test whether $G[U_i]$ is cobipartite, with $U_i = A_i \cup B_i$, where every vertex $v \in A_i$ has a nonneighbour in B_i or a neighbour in $\bar{U}_i$, and $B'_i = N(B_i) \cap \bar{U}_i$ is a (possibly empty) clique complete to B_i .
 - Otherwise can we move v from A_i to B_i , minimising the size of A_i . Note that we can have $B_i = \emptyset$ if $G[U_i]$ is a clique.
- (5) If so, test whether $G[\bar{U}_i]$ is a cluster graph. If so, $\mathbf{h} \leftarrow A_i, \mathbf{b}_i \leftarrow B_i \cup B'_i$ and the satellites are given by $\mathbf{b}_i$ and $G[\bar{U}_i \setminus B'_i]$. Stop, we have a unipolar decomposition.
 - U_i is a cobipartite separator in G . It cannot have vertices in two satellites, or else it is not cobipartite. Also $\mathbf{h} \subseteq U_i$ or $G[\bar{U}_i]$ would not be a cluster graph. So $U_i \subseteq \mathbf{h} \cup \mathbf{b}_i$. Now if $u_i \in \mathbf{h}, \mathbf{h} \subseteq U_i$, so $U_i = \mathbf{h} \cup B_i$, where by (*) we must have $A_i = \mathbf{h}$ and $B_i \subseteq \mathbf{b}_i$. Since B'_i is a clique, $G[\bar{U}_i]$ is a cluster graph if $B'_i \neq \emptyset$. But we can expand B_i to $\mathbf{b}_i$ by adding B'_i , since it is complete to B_i . Now $G[\bar{U}_i \setminus B'_i]$ gives the other satellites $\mathbf{b}_j$ ($j \in [s \setminus i]$). If $u_i \in \mathbf{b}_i, \mathbf{h} \subseteq U_i$ implies that u_i is complete to both $\mathbf{h}$ and $\mathbf{b}_i$, and we can move u_i into $\mathbf{h}$, and follow the reasoning above with $B'_i = \emptyset$.
- (6) Set $U_i \leftarrow w_i$ and reset quantities as in step (3).
 - $G[U_i]$ is a general unipolar subgraph of G , so $u_i \in \mathbf{h}$. Any nonedge in G has at least one vertex in a satellite, since $\mathbf{h}$ is a clique. So $u_i \in \mathbf{h}$ implies $w_i \in \mathbf{b}_i$, and $u_i \leftarrow w_i$ implies $G[U_i]$ is a subgraph of $\mathbf{h} \cup \mathbf{b}_i$.
- (7) For each $v \in U_i$, let $W(v) \leftarrow N(v) \cap \bar{U}_i$ and determine whether $W(v)$ is a clique. If not, we set $U_i \leftarrow U_i \setminus v$, removing v from U_i .
 - If $v \in \mathbf{b}_i, W(v) \subseteq \mathbf{h}$, so is a clique. Thus any v which is removed from U_i must be in $\mathbf{h}$, not in $\mathbf{b}_i$. Any remaining $v \in U_i \cap \mathbf{h}$ must be such that $W(v) \cup U_i = \mathbf{h} \cup B_i$, where $B_i \subseteq \mathbf{b}_i$, since $v \in \mathbf{h}$. Note that $\mathbf{h}$ here may depend on v unless $\mathbf{h}$ is unique.
- (8) If U_i is a clique and $N(U_i)$ is a clique, set $\mathbf{b}_i \leftarrow U_i$ and $V_{i+1} \leftarrow V_i \setminus U_i$. Let $i \leftarrow i + 1$ and return to step (2).
 - This will be true if and only if, $U_i \cap \mathbf{h} = \emptyset$, in which case $U_i = \mathbf{b}_i$.
- (9) Determine a $v \in U_i$, such that $W(v) \leftarrow W(v) \cup U_i$ gives a unipolar decomposition as in steps (4) and (5) above. If so, stop with this unipolar decomposition.
 - Then $W(v) = \mathbf{h} \cup B_i$, where $B_i \subseteq \mathbf{b}_i$, so $W(v)$ is a cobipartite separator in G as in steps (4) and (6).
- (10) Stop, G is not a unipolar graph.
 - Since we have not stopped with a unipolar decomposition, G is not unipolar.

It is not difficult to see that UNIPOLAR has an $\mathcal{O}(ms)$ time bound, where s is the number of satellites, since we consider at most $2s$ vertices u_i, w_i . For each such vertex, the most time-consuming operation are the $\mathcal{O}(m)$ time cobipartite decompositions. Since $s < n$, this is clearly an $\mathcal{O}(mn)$ time algorithm. However it appears that the time complexity can be reduced using a more careful implementation and analysis, possibly to $\mathcal{O}(m)$. See [38] for a different algorithm. For example, if we find $v \in \mathbf{h}$ during the identification of a satellite, this information can be used when identifying further satellites. But we will not pursue this here.

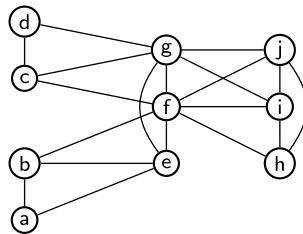
As an example, consider the graph in Fig. 14. Suppose we choose nonedge $u_1 w_1 \leftarrow gh$, giving $U_1 = N[g] \leftarrow \{c, d, e, f, g, i, j\}$. Then $G[U_1]$ is not cobipartite, so $g \in \mathbf{h}$ and $u_1 \leftarrow h$. Then $U_1 = N[h] \leftarrow \{f, h, i, j\}$. Now f has nonedge neighbours in U_1 , c, d for example, so $U_1 \leftarrow \{h, i, j\}$ which has neighbours only in the clique $\{f, g\}$ in $\bar{U}_1$. So $\mathbf{b}_1 \leftarrow \{h, i, j\}$ and $f, g \in \mathbf{h}$. Now $V_2 \leftarrow \{a, b, c, d, e, f, g\}$. Suppose now $u_2 w_2 \leftarrow ed$, so $U_2 \leftarrow \{a, b, e, f, g\}$. Now $G[U_2]$ is cobipartite, with $A_2 \leftarrow \{e, f, g\}$ and $B_2 \leftarrow \{a, b\}$, and $G[\bar{U}_2]$ is a cluster graph with cliques $\{c, d\}, \{h, i, j\}$. Now $B'_2 = \emptyset$, so $\mathbf{h} \leftarrow \{e, f, g\}, \mathbf{b}_1 \leftarrow \{h, i, j\}, \mathbf{b}_2 \leftarrow \{a, b\}$ and $\mathbf{b}_3 \leftarrow \{c, d\}$ is a unipolar decomposition.

4.3. Identifying a node

A sufficient condition for $\mathbf{u}$ to be a node in a thick tree G is that it $\mathbf{u}$ is a clique and $G[V \setminus \mathbf{u}]$ is a thick forest with s trees rooted at $\mathbf{u}$'s neighbours $\mathbf{w}_i$ ($i \in [s]$). But to find such $\mathbf{u}$ we use weaker necessary conditions: that $G[N[\mathbf{u}]]$ is a unipolar graph with satellites $C_i \subseteq \mathbf{w}_i$, and that the C_i are not connected in G .

So, at a general step, we have a clique $C \subseteq \mathbf{u}$ for some node $\mathbf{u}$ in G and we wish to find $\mathbf{u}$. To initialise the process, we may choose any $v \in V$. This must be in some node $\mathbf{u}$. Then $C = \{v\} \subseteq \mathbf{u}$, as required.

Since $C \subseteq \mathbf{u}$, we construct the graph $G_C = G[V_C]$, where $V_C = N[C]$. Now any clique in G containing C is also a clique in G_C and so $\mathbf{u} \subseteq V_C$. Thus G_C is a unipolar subgraph of $G_{\mathbf{u}}$ with hub $\mathbf{h}$ and satellites $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_s$, where $\mathbf{b}_i$ is contained in a thick neighbour $\mathbf{w}_i$ of $\mathbf{h}$, for $i \in [r]$. Also, since C is a hub for G_C , and $C \subseteq \mathbf{u}$, and $\mathbf{u} \subseteq V_C$, $\mathbf{u}$ is a hub for G_C .

Fig. 14. Unipolar graph G .

Note that $\mathbf{u}$ is a clique separator in G if $\mathbf{u}$ is not a thick leaf, and hence also in G_C . Thus any clique in G_C which includes $\mathbf{u}$ is a clique separator in G . However, G_C also has vertices from neighbours w_i ($i \in [s]$) of $\mathbf{u}$ in $H = \text{thin}(G)$, where $s \leq r = \deg_H(\mathbf{u})$. Since w_i is a clique, these also induce cliques in G_C . Also $\mathbf{u}$ may separate G into two graphs, $G_L = G[V_L]$ and $G_R = G[V_R]$, connected by $\mathbf{u}$. We will take $V_C \subseteq V_L$ and $V_R = V \setminus V_L$. Thus $V_L \neq \emptyset$, but $V_R = \emptyset$ is possible if $C \neq \mathbf{u}$. Now $V_L \setminus V_C$ can be discovered by searching in G starting from $N(C)$, and hence V_R can be identified. This preprocessing requires $\mathcal{O}(m)$ time.

We apply the algorithm UNIPOLAR to G_C , giving a hub $\mathbf{h}$ such that $C \subseteq \mathbf{h} \subseteq \mathbf{u} \subseteq V_C$. The only modification is that a vertex v cannot be in a satellite if it has a neighbour in V_R or is in C . This is required for $\text{thin}(G)$ to be triangle-free and to ensure that $C \subseteq \mathbf{h}$. Then, at termination, $\mathbf{h}$ is a hub for G_C and the only edges from V_L to V_R are from $\mathbf{h}$. Thus, if we check that $G[N(\mathbf{h})]$ is a cluster graph with cliques C_i ($i \in [s]$), we can take $\mathbf{u} \leftarrow \mathbf{h}$. We check that all the C_i are cliques in $\mathcal{O}(m)$ time, and are disconnected in G , which can be done with a breadth-first search starting from the C_i in $\mathcal{O}(m)$ time. The time needed to identify $\mathbf{u}$ is $\mathcal{O}(mn)$, from the time bound for determining V_L, V_R , the time bound for UNIPOLAR and the time bound for checking that $N(\mathbf{u})$ is a cluster graph. This is dominated by UNIPOLAR.

Note that we will only obtain a model in case (a) of Lemma 7 from the way G_C is constructed. Thus, if G is chordal, we claim $\mathbf{u} \leftarrow \mathbf{h}$, since two satellites in G_C cannot be connected via a vertex not in V_C . This case was considered by Stacho [41]. Otherwise G contains a hole of size at least four, so is not chordal. But the existence of 4-holes in G can mean that the hub $\mathbf{h}$ from UNIPOLAR satisfies only $\mathbf{h} \subset \mathbf{u}$. That is, we are in case (b) of Lemma 7. We must deal with this below, but first consider the chordal graph of Fig. 7 as an example, taking $C = \{z\}$. Thus $V_C = \{t, v, x, y, z\}$, $V_R = \emptyset$ and G_C comprises three triangles with common edge yz . So UNIPOLAR gives $\mathbf{h} = \{y, z\}$ and satellites $\{t\}$, $\{v\}$ and $\{x\}$. Then $N(\mathbf{h})$ gives $C_1 = \{s, t\}$, $C_2 = \{u, v\}$ and $C_3 = \{w, x\}$. These are disconnected in G , and so we may take $\mathbf{u} = \{y, z\}$. Continuing this into the three subtrees headed by the C_i ($i \in [3]$) gives the expected horizontal K_2 's as a further 12 nodes.

Now, in the presence of 4-holes, two of the satellites produced by UNIPOLAR may be connected as in Fig. 6 of Lemma 7. Then we cannot claim $\mathbf{u} \leftarrow \mathbf{h}$. However, since 4-holes can only occur in the links of a thick tree, two of the vertices of any 4-hole must be in $\mathbf{u}$ and two must be in some w_i . Thus the paths connecting satellites in G_C must be P_3 's having one endpoint in a satellite $\mathbf{b}_k \subseteq \mathbf{u} \setminus \mathbf{h}$, and the other in some $\mathbf{b}_j$ ($j \neq k$), with a mid vertex not in V_C . If the shortest path is not a P_3 , G has a hole of size 5 or more, so can be rejected.

So we search for these P_3 's, using a breadth-first search starting from the $\mathbf{b}_i \subseteq V_C$ given by UNIPOLAR, in $\mathcal{O}(m)$ time. We could now simply add these paths to G_C , giving a graph G'_C , and repeat UNIPOLAR on G'_C , but there is a more efficient alternative.

We sort the endpoints of the P_3 's by the $\mathbf{b}_i$ in which they appear. If at least three $\mathbf{b}_i$'s contain endpoints, this must identify a unique $\mathbf{b}_k$ such that $\mathbf{h}' = \mathbf{h} \cup \mathbf{b}_k$ is a node. Otherwise, G is not a thick tree since $\text{thin}(G)$ contains a 4-cycle which is not in a link. Now we test whether $N(\mathbf{h}')$ induces a cluster graph in G . If not, $N(\mathbf{h}')$ is not unipolar, so G is not a thick tree. If only two $\mathbf{b}_i$'s contain endpoints, there are two alternatives for $\mathbf{b}_k$, so we simply try both as above. If either succeeds we continue with this $\mathbf{u} \leftarrow \mathbf{h}'$. If both fail, again G is not a thick tree.

As an illustration, consider the graph of Fig. 15 with $C = b$. Then G_C is as shown, and UNIPOLAR gives $\mathbf{h} = \{b\}$ and satellites $\mathbf{b}_1 = \{a\}$, $\mathbf{b}_2 = \{c\}$ and $\mathbf{b}_3 = \{e\}$. These are connected in G by P_3 's ade and cfe . Both paths have endpoint e in $\mathbf{b}_3$ so we have $\mathbf{u} \subseteq \mathbf{h} \cup \mathbf{b}_3 = \{b, e\}$. Now e can be moved into $\mathbf{h}$, giving $\mathbf{h} \leftarrow \{b, e\}$. Now $N(\mathbf{h}) = \{a, d\} \cup \{c, f\}$ induce a cluster graph in G .

However, suppose, we had chosen $C = a$. Then G_C is the P_3 dab . Then UNIPOLAR gives $\mathbf{h} = \{a\}$ and satellites $\mathbf{b}_1 = \{b\}$, $\mathbf{b}_2 = \{d\}$. These are connected in G by a P_3 , bed , which has endpoints $b \in \mathbf{b}_1$ and $d \in \mathbf{b}_2$. So we have two candidates $\mathbf{u}_1 \subseteq \mathbf{h} \cup \mathbf{b}_1 = \{a, b\}$ and $\mathbf{u}_2 \subseteq \mathbf{h} \cup \mathbf{b}_2 = \{a, d\}$. Now $N(\mathbf{u}_1) = \{d, e\} \cup \{c\}$, which is not a cluster graph in G , since it is connected by the path cfe . Thus $\mathbf{u} \neq \mathbf{u}_1$, so we try $\mathbf{u}_2 = \{a, d\}$. Then $N(\mathbf{u}_2) = \{b, e\} \cup \{g\}$, which induces a cluster graph in G , since its two components are not connected. So we may take $\mathbf{u} \leftarrow \mathbf{u}_2$ and continue the algorithm.

We now take $C_i \subset w_i$ for the components C_i ($i \in [r]$) of the cluster graph $G[N(\mathbf{u})]$ and use the same process to recognise the r subtrees rooted at the w_i , terminating when all subtrees are cliques.

4.4. Fixing a path

There is a remaining difficulty with the recognition algorithm. As described above, it could result in an incorrect rejection. By (*), the algorithm of Section 4.2 moves as many vertices as possible into one of its satellites. Thus we may

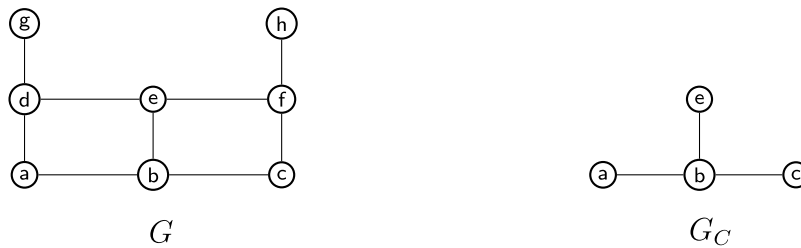


Fig. 15. Graph G with G_C for $C = \{b\}$.

mistakenly locate vertices from a node in one of its children and this cannot be recognised at that point in the algorithm. Hence we can misidentify thick vertices and edges, which can lead to further misidentifications.

The generic situation is shown in Fig. 16 by the loose triangle (abc) . As in Section 2.1, the heavy lines indicate full links, and the others are arbitrary links. Thus ab and bc are possible thick vertices, and (abc) represents a thick edge, which may be $ab : c$ or $a : bc$, as discussed in Section 2.1. We may also have loose 4-cycles, like $(efhg)$ in Fig. 16, with two full links representing the thick vertices of a thick edge $ef : gh$. As in Section 2.1, the 4-cycle can have chords, but we assume these links are not full, or the 4-cycle represents two thick edges. There is no need consider longer cycles since thick edges contain only triangles and 4-holes.

A loose triangle with exactly one full link can be a thick edge in one way, so there can be no misidentification. If there are two full links, the triangle can be a thick edge in two ways, and if there are three, the triangle is a thick vertex.

Thus confusion between thick vertices and edges can occur. We can view this as vertices being moved wrongly from a thick vertex into one of its neighbours. This can then propagate into one of the subtrees, and only one, by Cor. 2. Thus misidentifications can only occur along a path in the thick tree. And, since the thin graph must be a tree, the path must terminate at or before a thick.

This becomes problematic only when one of the nodes of the loose triangle is also in a loose 4-cycle, as with $(efhg)$ in Fig. 16. Here we must identify $de : gh$ as a thick edge or the graph is not a thick tree. However, this can result in a node being contained in two thick vertices, so we do not have a thick tree. We will call this a *clash*. In Fig. 16, for example, e is in both de and ef , so is a clash.

To resolve this, we must remove the clashing node from one of the thick vertices. In Fig. 16 we cannot remove e from ef or we have a 4-cycle which is not in a thick edge, implying rejection of G . So we must remove d from de , if possible. We can do this here since cd is a full link, giving $cd : e$ a thick edge, and de no longer a thick vertex. We identify cd as follows. Let N^- denote neighbourhood upwards in the tree. Since $N^-(e) = c \cup d$, we must move d into the thick vertex cd . If cd is not a full link, then G is not a thick tree, since we cannot resolve the clash. But resolving this clash may now cause a clash with thick vertex bc . Removing c from bc will then give the thick vertex ab and so on. The path of nodes and links involved is identified by the algorithm of Section 4. This path of clashes cannot go beyond the root, since that is where the first misidentification could occur. So this is a kind of backtracking but, since there is no branching, it does not significantly worsen the time complexity of the algorithm.

We accept G if this terminates with no clash, and reject G if there is a clash which cannot be resolved because some upward neighbourhood is not a clique. Encountering a 4-cycle now results in rejection. We cannot move the clashing node upwards in the schematic, since its neighbourhood is a P_3 . This occurs if both nodes in a thick vertex have links to a third. For example, if there was a node k in Fig. 16 so that bck formed a triangle, making cd a thick vertex gives a fatal clash at c . In fact, such a fatal clash reveals a forbidden subgraph from the class depicted in Fig. 12.

This problem can occur on several paths in the thick tree, even if G is accepted. But each such path is bounded by different loose triangles and 4-cycles in the schematic, so these paths must all be disjoint. Thus the total work to perform all fixes is $\mathcal{O}(m)$.

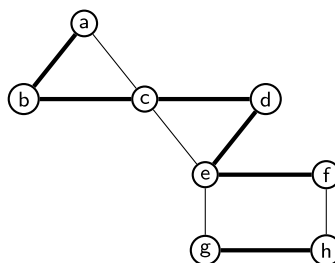


Fig. 16. Fixing a path.

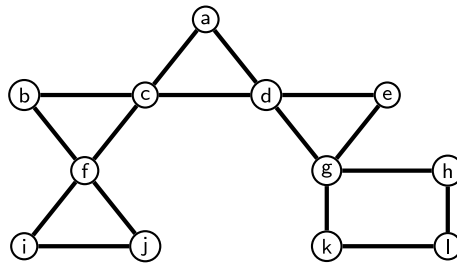


Fig. 17. Example: Fixing a path.

To illustrate this, consider the simple graph of Fig. 17. The vertices are 1-cliques, and therefore all links are full, as shown.

Suppose we start with $C = \{a\}$, then UNIPOLAR gives $\mathbf{u} = \{a\}$, and $C_1 = \mathbf{w}_1 = cd$. The node cd then has children bf and eg . In the subtree rooted at bf , we have a leaf ij , so this subtree is accepted. The node eg has two children h and k but these are connected by the P_3 klh . The 4-cycle $gklh$ can be resolved as a thick edge $gh : kl$ or $gk : hl$, but now g is a clash with eg . So we must move g . Now $N^-(g) = d \cup e$, which can be a node de . But now d clashes with cd . So we must move d from cd to give a new root ac , and we are done. So we accept G as a thick tree with nodes, ac , bf , ij , de , gh and kl . The path of modifications changes the nodes a , cd , eg to ac , de , gh .

4.5. Running time analysis

We now determine the time and space complexity of the algorithm. The space required is clearly $\mathcal{O}(m)$. The time to identify a node is $\mathcal{O}(mn)$. We claim that the recognition algorithm has time complexity $T(m, n) = \mathcal{O}(mn)$. This is easily seen to be true for the nodes. The recursion then splits G into subtrees, which we will assume have n_i vertices and m_i edges ($i \in [r]$).

Then, since $m_i + n_i < m + n$ for $i \in [r]$, the time complexity is bounded by

$$\begin{aligned}
 T(m, n) &= \sum_{i=1}^r T(m_i, n_i) + \mathcal{O}(mn), \text{ to determine a root,} \\
 &= \mathcal{O}\left(\sum_{i=1}^r m_i n_i\right) + \mathcal{O}(mn), \text{ by induction,} \\
 &= \mathcal{O}\left(\sum_{i=1}^r m_i \sum_{j=1}^r n_j + mn\right) \\
 &= \mathcal{O}(mn), \text{ since } \sum_{i=1}^r m_i \leq m, \sum_{j=1}^r n_j \leq n.
 \end{aligned}$$

Thus the overall time complexity of recognising a thick forest is $\mathcal{O}(mn)$.

5. Clique cutset decomposition

An important property of thick forests is that they have a *clique cutset decomposition*, given by the internal nodes. In general, a clique cutset decomposition gives a tree whose internal vertices correspond to clique separators and leaves called *atoms*, which are either cliques or subgraphs with no clique separator. See [4,7] for definitions and some applications of clique cutset decomposition, and [7,42] for efficient algorithms. Using this algorithm, the clique cutset decomposition of any graph G can be done in $\mathcal{O}(mn)$ time, the same as our time bound for recognising a thick forest.

It is easy to see that clique cutset decompositions have the following property.

Proposition 1. *Let $\mathcal{C}$ be a hereditary graph class, and let $\mathcal{C}_0$ be its subclass of graphs which are either cliques or have no clique separator. Then, in any clique cutset decomposition of $G \in \mathcal{C}$, all atoms must be in $\mathcal{C}_0$.*

Note that $\mathcal{C}_0$ is not hereditary in general, since removing a vertex can produce a graph with a clique separator. For example, if $\mathcal{C}_0$ is the class of long holes, removing any vertex gives a path which has a clique separator.

For a thick tree all atoms are either cliques or cobipartite graphs. However, clique cutset decomposition does not necessarily recover the thick tree structure, for two reasons. Firstly, a thick tree may have clique separators which are not nodes. Secondly, the cliques in a clique cutset decomposition do not need to be disjoint, as they do in a thick tree.

Thus we cannot use clique cutset decomposition to recognise a thick forest. However, the fact that it decomposes the graph is very useful in algorithmic applications, since it can reduce the problem on $\mathcal{C}$ to that on $\mathcal{C}_0$. For example, we will use it in sections 6.1 and 6.2 to give algorithms counting independent sets and colourings for graphs in $\mathcal{F}$.

This leads us define a larger graph class $\mathcal{Q}$ by simply requiring that all atoms of a clique cutset decomposition are either cliques or cobipartite graphs with no clique separator. We will call $\mathcal{Q}$ the class of *quasi* thick forests. Clearly $\text{thick}(\mathcal{F}) \subseteq \mathcal{Q}$. But, from Proposition 1, we see that if $G \in \mathcal{Q}$, then any clique cutset decomposition of G proves membership in the class, which is not true for $\mathcal{F}$. A chordal thick edge is a cochain graph, which decomposes into two overlapping cliques, which is not permitted for a thick tree.

A graph in $\mathcal{Q}$ can be recognised in $\mathcal{O}(mn)$ time using clique cutset decomposition and the recognition of cobipartite graphs from Section 4.1. Quasi thick forests include the class $\mathcal{C}_\Delta$ of chordal graphs, where the atoms of the clique cutset decomposition are all cliques. A graph is chordal if and only if it has a clique cutset decomposition with only cliques as atoms [28].

Clearly not all thick forests are chordal, but neither are all chordal graphs thick forests. The graph shown in Fig. 8 is an example of a chordal graph which is not a thick tree, as we proved in Section 3.2. Thus $\mathcal{F} \cup \mathcal{C}_\Delta \subseteq \mathcal{Q}$. A non-chordal quasi thick tree may also fail to be a thick tree. See Fig. 9 for an example. This graph is in $\mathcal{Q}$ since the clique separator $\{a\}$ decomposes it into two cobipartite graphs $abfd$ and $acge$, but we proved in Section 3.2 that it is not in $\mathcal{F}$. Thus $\mathcal{F} \cup \mathcal{C}_\Delta \subset \mathcal{Q}$.

We can use properties of clique cutset decomposition to prove facts about $\mathcal{C}$ from those for $\mathcal{C}_0$. These are stated without proof in [4], but follow easily from Proposition 1.

Proposition 2. *Let H be a graph with no clique separator. Then any induced copy of H in a graph G must be contained in an atom of any clique cutset decomposition of G .*

We can use Proposition 2 to prove perfection of $\mathcal{Q}$.

Proposition 3. *A hole or (long) antihole has no clique separator.*

Proof. Clearly a hole has no clique separator. So let the antihole be $\bar{C}$, where $|C| = k \geq 5$. Any vertex in $\bar{C}$ has only two non-neighbours, so any clique separator $S \subset \bar{C}$ must separate one vertex v from at most two others. Thus $|S| \geq 2$ and we must have $N(v) \subseteq S$. But $N(v)$ induces a P_{k-2} in C , so $\bar{C}[S]$ has at least one non-edge, a contradiction. $\square$

The following is then a direct consequence of the SPGT and Props. 2 and 3.

Lemma 13. *A graph is perfect if and only if all atoms of its clique cutset decomposition are perfect.* $\square$

Cobipartite graphs can contain even antiholes. In fact, an even antihole $\bar{C}_{2k}$ is a connected cobipartite graph for any $k \geq 3$.

Thus $\mathcal{Q} \subseteq \mathcal{P}$, but $\mathcal{Q}$ has a stronger inclusion. Recall that $\mathcal{P}_0$ is the class of long hole-free perfect graphs.

Lemma 14. $\mathcal{Q} \subset \mathcal{P}_0$.

Proof. $\mathcal{Q} \subseteq \mathcal{P}_0$ follows from Proposition 1 and that cobipartite graphs are in $\mathcal{Q} \subseteq \mathcal{P}_0$.

To see that $\mathcal{Q} \neq \mathcal{P}_0$, note that otherwise any atom in the clique cutset decomposition of $G \in \mathcal{P}_0$ must be cobipartite. Thus we need only exhibit possible atoms which are neither. Complete bipartite graphs K_{ij} with $i \geq 2, j \geq 3$ give an example. These are perfect, have no holes and no clique separators, yet are not cobipartite. $\square$

Thus the first of the forbidden subgraphs for thick forests in Fig. 11 is also a forbidden subgraph for $\mathcal{Q}$, since it is $K_{2,3}$. But none of the other graphs in Figs. 11 and 12 are forbidden for $\mathcal{Q}$, since they are chordal and/or have cut edges.

6. Counting independent sets and colourings

6.1. Counting independent sets in $\mathcal{Q}$

We consider counting weighted independent sets, meaning evaluating the weighted independence polynomial. The vertices $v \in V(G)$ have non-negative weights $w(v) \in \mathbb{Q}$. Let $\mathcal{I}_k(G)$ denote the independent sets of size k in G , and $\alpha(G) = \max_k \{|\mathcal{I}_k(G)| \neq \emptyset\}$. The weight of a subset S of V is defined to be $w(S) = \prod_{v \in S} w(v)$. Then let $W_k(G) = \sum_{S \in \mathcal{I}_k(G)} w(S)$, and $W(G) = \sum_{S \in \mathcal{I}(G)} w(S) = \sum_{k=0}^n W_k(G)$. The weighted independence polynomial is $P(\lambda) = \sum_{k=0}^n W_k \lambda^k$, with the W_k as its coefficients, so $P(\lambda)$ can be regarded as $W(G)$ with weights $\lambda w(v)$.

The following is proved in [20, Sec. 2.2].

Theorem 2. *Let $\mathcal{C}$ be a hereditary class of graphs such that every graph in $\mathcal{C}$ has a clique cutset decomposition with all atoms in the class $\mathcal{C}_0 \subseteq \mathcal{C}$. Suppose we can evaluate $W(G)$ for any $G \in \mathcal{C}_0$ in time $T_0(n)$, assumed $\Omega(n)$ and convex. Then we can evaluate $W(G)$ for any $G \in \mathcal{C}$ in time $T(n) \leq 2nT_0(n)$.*

Here *evaluation* means either exact computation in polynomial time or approximate computation using an FPRAS [31]. Error control for approximation is described in [20, Sec. 2.2.1].

The paper [20] is concerned with evaluating $W(G)$ in claw- and fork-free perfect graphs. Clique cutset decomposition is used for claw-free perfect graphs, where the atoms are either graphs with small independence number or (essentially) line graphs of bipartite graphs, where $W(G)$ can be evaluated using the algorithm of [20]. The extension to fork-free graphs uses a different technique, *modular decomposition*, which we will not discuss here.

Thick forests are perfect, as we have seen, but are not claw-free or fork-free in general. A thick tree G can contain a claw, possibly many, unless the thin tree H is a path, and can contain a fork unless the thin tree H is a star, that is, G is unipolar. Thus thick forests are perfect graphs, but not necessarily claw-free or fork-free, so are not within the class of graphs considered in [20]. Note that a claw or a fork, in fact any tree, does not fall within the scope of Proposition 2 since they have clique separators of size one.

We can use Theorem 2 to evaluate $W(G)$ for $G \in \mathcal{Q}$, implying the same result for thick($\mathcal{F}$). The difference from [20] is in the atoms of the clique cutset decomposition. Here they are simply cliques and cobipartite graphs. For these we can evaluate the W_k exactly in linear time. For a clique K , $W_0 = 1$, $W_1 = \sum_{v \in K} w(v)$, $W_k = 0$ ($k > 1$). For a cobipartite graph H , $W_0 = 1$, $W_1 = \sum_{v \in H} w(v)$, $W_2 = \sum_{uv \notin E(H)} w(u)w(v)$ and $W_k = 0$ ($k > 2$). Thus we can evaluate $W(G)$ exactly for $G \in \mathcal{Q}$, and there is no need for the error control. Thus we have a deterministic algorithm for counting weighted independent sets in $\mathcal{Q}$.

A consequence of exact evaluation of $W(G)$ is that we can evaluate $W_k(G)$ exactly for all $0 \leq k \leq n$ by interpolation. We can evaluate $P(\lambda)$ by modifying the weights $w(v)$ to $\lambda w(v)$. Since $W_0 = 1$, if we do this for $\alpha(G)$ positive values of λ , we can solve α linear equations for the coefficients W_k ($k \in [\alpha]$).

6.2. Counting colourings in $\mathcal{Q}$

We will consider evaluating $\#col_q(G)$ for $G \in \mathcal{Q}$ using clique cutset decomposition in a similar way Theorem 2.

The proof of Theorem 2 given in [20] remains valid for any quantity if a clique cutset $G_1 \cap G_2$ can be used to lift evaluations of the quantity from G_1 and G_2 to their union $G = G_1 \cup G_2$. For $W(G)$ this uses the simple fact that any clique can only contain one vertex of an independent set. To evaluate $\#col_q(G)$ we may replace this by the following *clique cutset colouring lemma* (CCCL).

Lemma 15 (CCCL). *Suppose $G = (V, E)$ and $K \subseteq V$ is a clique separator of size $k = |K|$ in G . Let $G_i = G[V_i]$ ($i \in [2]$), where $K = V_1 \cap V_2$. Then $\#col_q(G) = \#col_q(G_1)\#col_q(G_2)/(q)_k$.*

Proof. K has $(q)_k$ colourings, each corresponding to permuting a selection of k of the q colours. Fix a colouring of K and count its extensions to colourings of G , G_1 and G_2 . The symmetry of q -colourings under permutation of colours implies that these numbers are the same for each colouring of K . However, in the product $\#col_q(G_1)\#col_q(G_2)$, each colouring of G is counted $(q)_k$ times, so we correct for this overcounting. We do not need to assume G is connected if we allow $\emptyset$ as a clique separator of size 0. $\square$

Note that the proof of the CCCL relies completely on the symmetries of both q -colouring and cliques. Thus it does have any obvious generalisations.

Thus, if we evaluate $\#col_q$ for all atoms of the clique cutset decomposition of G , we can use the CCCL to evaluate $\#col_q(G)$. This gives the following analogue of Theorem 2.

Theorem 3. *Let $\mathcal{C}$ be a hereditary class of graphs such that every graph in $\mathcal{C}$ has a clique cutset decomposition with all atoms in the class $\mathcal{C}_0 \subseteq \mathcal{C}$. Suppose we can evaluate $\#col_q(G)$ for any $G \in \mathcal{C}_0$ in time $T_0(n)$, assumed $\Omega(n)$ and convex. Then we can evaluate $\#col_q(G)$ for any $G \in \mathcal{C}$ in time $T(n) \leq 2nT_0(n)$.*

In [20] colourings could not be counted since evaluation of $\#col_q$ in the atoms that are line graphs of bipartite graphs is not possible in general. That is, we cannot count edge-colourings of bipartite graphs, even approximately.

To apply Theorem 3 to $\mathcal{Q}$, we must consider evaluation of $\#col_q(G)$ when G is a clique or cobipartite graph. The number of colourings of a clique of size k is exactly $(q)_k$ from above. Note that if G is a chordal graph, there are no cobipartite atoms, so we have an exact algorithm for counting q -colourings of chordal graphs. We could implement this to count colourings in chordal graphs in $\mathcal{O}(m + n)$ time, but we note this can be done more easily using the algorithm of [1, Rem. 2.5].

By contrast, we show below that exact counting is $\#P$ -complete for cobipartite graphs. So we can only count colourings approximately. If we can approximate $\#col_q$ for cobipartite graphs, we can use the CCCL to approximate $\#col_q(G)$. We use the following.

Lemma 16. *Let G be a cobipartite graph on n vertices. Let $B = \bar{G}$ and let $\ddot{B}$ be its bipartite complement. Let $\kappa_k(\ddot{B})$ be the number matchings of size k in $\ddot{B}$. Then*

$$\#col_q(G) = \sum_{k=n-q}^n \kappa_k(\ddot{B})(q)_{n-k}.$$

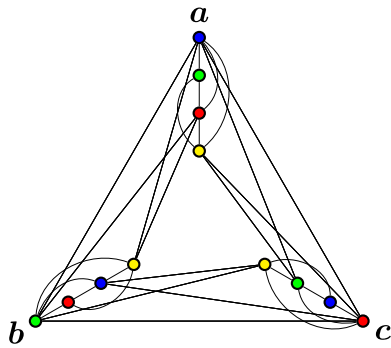


Fig. 18. A proper 4-colouring of a thick triangle with K_4 nodes.

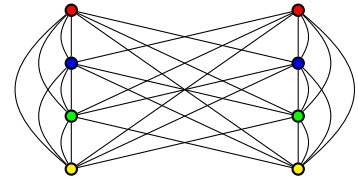


Fig. 19. A copy link with $q = 4$.

Proof. Consider q -colouring a cobipartite graph G on $n = n_1 + n_2$ vertices, having cliques $C_i = (V_i, E_i)$ ($|V_i| = n_i$, $i \in [2]$), connected by a bipartite graph $B = (V_1 \cup V_2, E_{12})$. Then C_i can be coloured in $(q)_{n_i}$ ways, since all vertices must be coloured differently. Also vertices $v_i \in V_i$ ($i \in [2]$) must be coloured differently if $\{v_1, v_2\} \in E_{12}$. Thus vertices in V_1, V_2 can only receive the same colour if they form a matching in $\bar{B}$, the bipartite complement of B . If $\bar{B}$ has a matching M of size k , G can be coloured in $(q)_{n-k}$ ways, by arbitrarily colouring all vertices not in M and one vertex from each edge in M . Thus, if $\bar{B}$ has κ_k matchings of size k , G has $\kappa_k(q)_{n-k}$ colourings corresponding to these. Summing over the possible values of k now gives the result. $\square$

An alternative proof of Lemma 16 can be given using the perfection of cobipartite graphs, but it is not simpler than the above.

The following is then an easy consequence of Lemma 16.

Lemma 17. Counting the number of q -colourings of a cobipartite graph is #P-complete.

Proof. If $n_1 = n_2 = q$, then $\kappa_k = 0$ for $k > q$ and $\#\text{col}_q(G) = \kappa_q q!$ from Lemma 16. So $\#\text{col}_q(G)/q!$ is the number of perfect matchings of $\bar{B}$. Thus we have a reduction to $\#\text{col}_q$ from counting the number of perfect matchings in a bipartite graph. Valiant [44] showed this to be #P-complete. $\square$

Thus we cannot hope for exact counting, but there is an FPRAS (fully polynomial randomised approximation scheme) to approximate the number of k -matchings in G to relative error ε with high probability [32]. Since Lemma 16 gives a linear function of κ_k with positive coefficients, we can use an FPRAS to approximate the κ_k to give an FPRAS to approximate $\#\text{col}_q(G)$ on any cobipartite graph for any q . We can then use Theorem 3 to give an FPRAS for $\#\text{col}_q(G)$ for any $G \in \mathcal{Q}$ and any q .

Finally, note that if the chromatic number $\chi(G)$ for $G \in \mathcal{Q}$ is a parameter, there is an easy FPT algorithm for counting colourings exactly in $\mathcal{Q}$. Since G is perfect χ is the size of the largest clique. Thus the largest clique in G has at most χ vertices, or else $\#\text{col}_q(G) = 0$. Therefore any atom of the clique cutset decomposition of G is either a clique of size at most χ or a cobipartite graph G_0 such that $\bar{G}_0$ has a bipartition with both parts of size at most χ . Then $\bar{G}_0$ has at most $\sum_{k=0}^{\chi} \binom{\chi}{k}^2 k! = \mathcal{O}(\chi^{2\chi})$ matchings of all sizes k , which we can count exactly by brute force. We can then use Lemma 16 to count the number of colourings of G_0 . Then we use Theorem 3 to extend the counting from the atoms to the whole of G . This algorithm has time complexity $\mathcal{O}(\chi^{2\chi} n)$ from Theorem 3, so is in FPT.

6.3. Counting colourings of a graph not in $\text{thick}(\mathcal{F})$

We will show that approximately counting proper colourings of a graph a thin graph which is not a forest is NP-hard. Thus the result of Section 6.2 is best possible.

First we show that

Lemma 18. If Δ is a triangle, it is NP-complete to determine if $T \in \text{thick}(\Delta)$ has a proper q -colouring.

Proof. We consider q -colouring a $T = (V, E) \in \text{thick}(\Delta)$ with three K_4 nodes a, b, c . (See Fig. 18.) Note that the nodes a, b, c are fixed and only the links ab, bc, ca are the input. Then each of the q colours must occur exactly once in each node. Observe now that T can be properly q -coloured if and only if the vertices of $\bar{T} = (V, \bar{E})$ can be partitioned into q induced triangles. But it is shown in [15, Prop. 5.1] that it is NP-complete to decide if $\bar{T}$ has such a partition, even if $\bar{T}$ is 6-regular or, equivalently, G is $(3q - 7)$ -regular. $\square$

We can extend this to arbitrary H using copy links. These are cobipartite graphs with both cliques of size q , and a bipartite graph which is a $K_{q,q}$ minus a matching. (See Fig. 19.) The only proper colourings of such a link are those which have the same colourings of its thick end vertices.

Lemma 19. *If $H \notin \mathcal{F}$, it is NP-complete to determine if $G \in \text{thick}(H)$ has a proper q -colouring.*

Proof. We construct $G \in \text{thick}(H)$ as follows. All nodes of G will be K_q 's. Let $T \in \text{thick}(\Delta)$ be a thick triangle with thick K_q vertices **a**, **b**, **c** and input links **ab**, **bc**, **ca**. We will label the vertices of H with a, b or c, and associate these labels with the nodes **a**, **b**, **c** of T .

If $H \notin \mathcal{F}$, it has a cycle C . Label any three successive vertices of C with a, b and c. Now label all other vertices in H with c. If an edge uv in H is labelled ab, let $\text{thick}(uv) \cong \mathbf{ab}$ in G . If an edge uv of H is labelled xc for $x \in \{a, b\}$, we let $\text{thick}(uv) \cong \mathbf{xc}$ in G . Finally if an edge uv of H is labelled cc, let $\text{thick}(uv)$ be a copy edge in G . Now it is clear that G has a proper q -colouring if and only if T has a proper q -colouring. $\square$

Thus we have the following. Note that the class $\mathcal{C}$ is not restricted to be hereditary.

Theorem 4. *If $H \notin \mathcal{F}$, then counting proper q -colourings of graphs in $\text{thick}(H)$ is complete for #P under AP-reducibility.*

Proof. Follows directly from Lemma 19 and [17, Thm. 1]. $\square$

7. Parameterisation

Thick forests are a tractable class, but this does not seem true for most classes, in the light of Lemma 3. However, we can consider fixed-parameter tractability, as we did for exactly counting colourings in $\mathcal{Q}$ in Section 6.2. Here we consider two possible parameterisations by properties of the thin graph.

7.1. Parameterising by the size of the thin graph

The most natural parameter for thick graphs is the size of the thin graph. Even so, we must restrict attention to triangle-free graphs, the class $\mathcal{T}$. Otherwise, the recognition problem is NP-complete [37]. Let us denote the class of triangle-free graphs with at most ν vertices by $\mathcal{T}_\nu = \mathcal{T} \cap \mathcal{G}_\nu$. Then, given a graph G , we consider deciding whether $G \in \text{thick}(\mathcal{T}_\nu)$ with ν as a parameter.

The case where $\mathcal{C} = \{H\}$ for a fixed graph H was resolved by MacGillivray and Yu [37]. They proved the following, which we recast in our own notation.

Theorem 5 (MacGillivray–Yu). *Let $\nu = |V(H)|$. If $H \in \mathcal{T}$, the decision problem $G \in \text{thick}(H)$ with parameter ν is in XP. Otherwise it is NP-complete.* $\square$

This shows NP-completeness of recognising $H \in \text{thin}(G)$ for $G \notin \text{thick}(\mathcal{T})$, but shows that there is a recognition algorithm if the parameter is $H \in \mathcal{T}$. The algorithm given in [37] has complexity $n^{\Omega(\nu)}$, so is only an XP algorithm. An obvious question is whether this can be improved to an FPT algorithm. We note that this has been done for thick bipartite graphs in [34], where an FPT algorithm is given for the stronger parameterisation that ν is the size of the smaller part of the bipartition. This gives a different generalisation of unipolar graphs from $\text{thick}(\mathcal{F})$.

Theorem 5 answers a slightly different question from deciding whether $G \in \text{thick}(\mathcal{T}_\nu)$, so we first show equivalence for fixed-parameter tractability.

Lemma 20. *Recognition of $G \in \text{thick}(H)$ with parameter H is in FPT if and only if recognising $G \in \text{thick}(\mathcal{T}_\nu)$ with parameter ν is in FPT.*

Proof. Suppose we have a FPT algorithm with parameter $H \in \mathcal{T}$ to decide whether $G \in \text{thick}(H)$, where $\nu = |V(H)|$. Thus we can test whether $G \in \text{thick}(H')$ for each of the graphs $H' \in \mathcal{T}_\nu$ using the given FPT algorithm. We simply generate all $2^{\nu(\nu-1)/2}$ graphs on ν vertices and reject those with triangles. This gives an FPT algorithm with parameter ν to decide if $G \in \text{thick}(\mathcal{T}_\nu)$.

Conversely, suppose we have an FPT algorithm with ν as parameter which returns $H' \in \text{thin}(G) \cap \mathcal{T}_\nu$. Since $G \in \text{thick}(\mathcal{T})$, its neighbourhoods are unipolar. Then vertices can only be moved into one of the identified neighbourhoods of any node **u**, from Cor. 1. It follows that H' is the same as H up to contracting full links and/or splitting nodes. So any other graph $H'' \in \text{thin}(G) \cap \mathcal{T}_\nu$ can be formed by contracting the links and/or splitting the nodes of G . We generate all such graphs H'' by performing sequences of valid splits and contractions.

There are at most $2^{|E(H')|} \leq 2^{\nu^2}$ ways of contracting full links of G . Since $\deg(\mathbf{u}) < \nu$ for $\mathbf{u} \in H'$, there are less than 2^ν ways of splitting **u** so that the neighbourhoods of its two parts are disjoint, and thus 2^{ν^2} overall. Thus we generate less than $2^{2\nu^2}$ graphs H'' . If $H'' \notin \mathcal{T}$ or $|V(H'')| \neq \nu$ then H'' can be discarded. For the remainder, we check if H'' is isomorphic to H . At worst, this requires comparing all $\nu!$ relabellings of H with H'' . This gives an FPT algorithm for recognising $G \in \text{thick}(H)$. $\square$

We now show that

Theorem 6. *Recognition of $G \in \text{thick}(\mathcal{T}_\nu)$ with parameter ν is in FPT.*

Proof. First we check whether all components of G are all cliques and cobipartite graphs, using the method of Section 4.1. If so, the thin graph H has only isolated vertices and edge. If so, we accept if and only if $H \in \mathcal{G}_\nu$.

Otherwise, we choose $u \in V$ and use the method for recognising a node $\mathbf{u}$ of $G \in \text{thick}(\mathcal{F})$ given in Section 4, which applies equally to $G \in \text{thick}(\mathcal{T})$. If this fails, we reject G and stop.

Now $N(\mathbf{u})$ may have vertices misplaced to one of its thick neighbours $\mathbf{w}_i$ for $i \in [d]$, where $d = \deg(\mathbf{u})$. In recognising $G \in \text{thick}(\mathcal{F})$ this misplacement is not problematic, since its effect can only propagate along a path in H . Then we can recover from the misplacement as described in Section 4.4. But if $H \notin \mathcal{F}$, these paths of misplacements may collide, and efficient recovery may not be possible. However, suppose we have misplacement into one of $\mathbf{u}$'s thick neighbours, $\mathbf{w}_j$ ($j \in [d]$).

Now $N(\mathbf{u})$ gives the edge sets $\mathbf{u} : \mathbf{w}_i$ for $i \in [d] \setminus j$ so deleting these from G gives a graph G'_j with $(d - 1)$ fewer links and one more node with degree at most 1. Now $G'_j \in \text{thick}(\mathcal{T}_\nu)$ if and only if $G \in \text{thick}(\mathcal{T}_\nu)$. We now repeat the above on G'_j . This cannot continue for ν iterations or all vertices in $\text{thin}(G'')$ for the resulting graph G'' have degree at most 1. Then $\text{thin}(G'')$ comprises only isolated links and vertices, and will be accepted at the start of the next iteration.

The problem is that we may not know j , so we must try all d possibilities, giving rise to d graphs $G'_1, \dots, G'_d$. Now $G \in \text{thick}(\mathcal{T})$ if and only if at least one of these is in $\text{thick}(\mathcal{T})$. Since $d < \nu$ and at most ν iterations are necessary, at most ν^ν graphs are generated before acceptance or rejection. Since the required computation on each such graph runs in time in $\mathcal{O}(mn)$, from Section 4, the algorithm is in FPT with parameter ν . $\square$

Corollary 5. *The graph operation of replacing single edges by paths is called subdivision. Since we can fix a path, using the method of Section 4.4, it follows that the algorithm of Theorem 6 gives polynomial time recognition of the class of graphs which are thick subdivisions of triangle-free graphs on ν vertices. In particular, this includes all thick cycles of length at least 4 since these are subdivisions of a thick cycle of length 4, which is in $\text{thick}(\mathcal{T}_4)$. $\square$*

Remark 2. In [34, Thm. 1.4], Theorem 6 is proved for the special case $G \in \text{thick}(K_{\nu_1, \nu_2})$ and parameter $\nu = \nu_1 + \nu_2$. In fact, [34] give a stronger result [34, Thm. 1.3]: there is an FPT algorithm when the parameter is ν_1 only. This clearly does not follow immediately from Theorem 6. We believe our methods can be adapted to give this improvement, but we will not pursue it further here.

The criterion of Theorem 5 extends to any finite class $\mathcal{C}$, but not to infinite classes, though we know from Lemma 1 that $\text{thick}(\mathcal{C})$ is not recognisable for any hereditary class $\mathcal{C}$ such that $\mathcal{C} \not\subseteq \mathcal{T}$. We have seen that recognition of thick forests $\text{thick}(\mathcal{F}) \subset \text{thick}(\mathcal{T})$ is in P, but for thick bipartite graphs $\text{thick}(\mathcal{B}) \subset \text{thick}(\mathcal{T})$ recognition is NP-complete. Neither is being cycle-free a necessary condition, from Cor. 5. We leave as an open question what properties of $\mathcal{C}$ are necessary and sufficient for $G \in \text{thick}(\mathcal{C})$ to be decidable in P.

As for algorithms on $\mathcal{T}_\nu$, counting independent sets in the class of Theorem 6 is in XP, since any of the ν nodes may contain at most one vertex of the independent set. Thus we can simply check all of the $\mathcal{O}(n^\nu)$ possibilities. Though we will not prove it here, this problem is $\#W[1]$ -complete. (See, for example, [34, Prop. 4.2] for a similar result.) Thus we cannot hope for an FPT algorithm.

Counting colourings, even approximately, in $G \in \text{thick}(\mathcal{T})_\nu$ is clearly not tractable unless $G \in \text{thick}(\mathcal{F})$, by Theorem 4.

7.2. Parameterising by treewidth

Here we consider parameterising by the *treewidth* of the thin graph. Treewidth measures how treelike a graph is, see Section 1.2.2. Let $\mathcal{W}_k$ be the class of graphs with treewidth at most k .

Treewidth is an important parameterisation of graphs, because of its application in algorithmic meta-theorems [14]. Our interest in $\mathcal{W}_k$ here is that the class $\mathcal{W}_1 = \mathcal{F}$, and we have shown that $\text{thick}(\mathcal{F})$ is a tractable class. So we ask which of our results above can be extended to $\text{thick}(\mathcal{W}_k)$ for small $k > 1$.

Unfortunately, the classes $\text{thick}(\mathcal{W}_k)$ already have an NP-complete recognition problem for $k > 1$ by Lemma 1, since $\mathcal{W}_k$ includes the triangle for all $k > 1$. Whether the class $\text{thick}(\mathcal{W}_2 \cap \mathcal{T})$ has polynomial time recognition we leave as an open question, but this seems possible, since $\mathcal{W}_2$ contains only series-parallel graphs. In particular, $\mathcal{W}_2 \cap \mathcal{T}$ contains all cycles C_ℓ with $\ell > 3$, which do have polynomial time recognition from Cor. 5.

Though we cannot recognise $G \in \text{thick}(\mathcal{W}_k)$ in general, we may still be able to use this characterisation of G for algorithmic purposes provided a decomposition of G with thin graph $H \in \mathcal{W}_k$ is given. This is true of many other NP-complete graph parameters, clique-width for example [33]. In this vein, we will show in Section 7.2.1 that we can count independent sets exactly in $\text{thick}(\mathcal{W}_k)$.

As for counting colourings, we know by Lemma 19 that this is intractable for $G \notin \text{thick}(\mathcal{F})$, that is, $G \notin \text{thick}(\mathcal{W}_1)$. So counting colourings in $\text{thick}(\mathcal{W}_k)$ is intractable for any $k > 1$.

7.2.1. Counting independent sets

Here we show that there is a deterministic algorithm for computing $\#ind(G)$, the number of independent sets of a thick graph G if we are given a model (H, ψ) with $tw(H) \leq k$ for some $k > 1$.

If $\alpha(G)$ is the size of the largest independent set in G , the algorithm is based on the following lemma.

Lemma 21. *Let $G = (V, E)$ be a graph with model (H, ψ) and let (I, b) be a tree decomposition of H of width k . For every $i \in I$ we have $\alpha(G[\psi^{-1}(b(i))]) \leq k + 1$.*

Proof. The bag $b(i)$ contains at most $k + 1$ vertices of H since the width of (I, b) is k . Each vertex $v \in b(i)$ corresponds to a clique $\psi^{-1}(v)$ of G by property 1 of the model (H, ψ) . $\square$

We can now show

Lemma 22. *Let $G = (V, E)$ be a graph with model (H, ψ) , where $H \in \mathcal{W}_k$. Then there is an $\mathcal{O}(n^{k+2})$ algorithm for counting all independent sets in G .*

Proof. In $\mathcal{O}(n^{k+2})$ time we can find a nice tree decomposition (T, b) of H of width k , using the algorithm of [3]. Let $T = (I, A)$ with root r . For each node $i \in I$ let $C(i)$ be the set of descendants of i in T . These are the nodes $j \in I$ such that i is on the path from j to r in T , including i itself. Our algorithm computes recursively, for every $i \in I$ and every independent set $S \subseteq \psi^{-1}(b(i))$ of G , the numbers $a(i, S)$ of independent sets of $G(i) = G[\psi^{-1}(\bigcup_{j \in C(i)} b(j))]$ that coincide with S on $\psi^{-1}(b(i))$. That is, $\#ind(G(i)) = \sum_S a(i, S)$, where we sum over all independent sets $S \subseteq \psi^{-1}(b(i))$ of $G(i)$. The values of $a(i, S)$ are computed from the leaves of T up to the root r as follows.

leaf For every leaf l of T we have $a(l, \emptyset) = 1$, because the empty set $\emptyset$ is the only independent set in the empty graph $(\emptyset, \emptyset)$.

introduce Let i be an introduce node with child j and let $w \in b(i) \setminus b(j)$. For every independent set $S \subseteq \psi^{-1}(b(j))$ and every vertex $v \in \psi^{-1}(w)$ such that $S \cup \{v\}$ is independent we have $a(i, S) = a(j, S)$ and $a(i, S \cup \{v\}) = a(j, S)$. The former is obvious. The later equality follows from property (2) of tree decompositions: Since w is the new vertex in $b(i)$, all neighbours of v in $G(i)$ are in $\psi^{-1}(b(i))$.

forget Let i be an forget node with child j and let $w \in b(j) \setminus b(i)$. For every independent set $S \subseteq \psi^{-1}(b(j))$ and every $v \in \psi^{-1}(w)$ we have $a(i, S) = a(j, S)$ if $v \notin S$ and $a(i, S \cup \{v\}) = a(j, S \setminus \{v\}) + a(j, S)$ if $v \in S$.

join Let i be a join node with children j and j' . For every independent set $S \subseteq \psi^{-1}(b(i))$ we have $a(i, S) = a(j, S) \cdot a(j', S)$, because there are no edges of H between vertices in $C(j) \setminus b(i)$ and $C(j') \setminus b(i)$, and consequently there are no edges on G between $\psi^{-1}(C(j) \setminus b(i))$ and $\psi^{-1}(C(j') \setminus b(i))$.

root G has $\#ind(G) = a(r, \emptyset)$ independent sets because $S \cap b(r) = \emptyset$ holds for all independent sets S of G .

For an n -vertex graph G we must compute a table of $\mathcal{O}(n^{k+2})$ values $a(i, S)$ because the tree T has linear size, and by Lemma 22 the independent set S chooses at most $k + 1$ vertices from the set $\psi^{-1}(b(i))$. Consequently, the algorithm runs in time $\mathcal{O}(n^{k+2})$. $\square$

The algorithm can be modified easily to deal with weighted independent sets.

Thus we have an XP algorithm provided we are given an $H \in \text{thin}(G)$ of treewidth at most k . We leave as an open question whether there is an FPT algorithm for counting independent sets in this setting.

Acknowledgement

We thank Mark Jerrum for useful inputs and comments on an earlier version of this paper.

Data availability

No data was used for the research described in the article.

References

- [1] G. Agnarsson, On chordal graphs and their chromatic polynomials, *Math. Scand.* 93 (2003) 240–246.
- [2] M. Albertson, R. Jamison, S. Hedetniemi, S. Locke, The subchromatic number of a graph, *Discrete Math.* 74 (1989) 33–49.
- [3] S. Arnborg, D. Corneil, A. Proskurowski, Complexity of finding embeddings in a k -tree, *SIAM J. Algebraic Discret. Methods* 8 (1987) 277–284.
- [4] A. Berry, R. Pogorelnik, G. Simonet, An introduction to clique minimal separator decomposition, *Algorithms* 3 (2010) 197–215.
- [5] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.* 25 (1996) 1305–1317.
- [6] H.L. Bodlaender, A partial k -arboretum of graphs with bounded treewidth, *Theoret. Comput. Sci.* 209 (1998) 1–45.
- [7] V. Boncompagni, I. Penev, K. Vušković, Clique-cutsets beyond chordal graphs, *J. Graph Theory* 91 (2019) 192–246.
- [8] D. Bovet, P. Crescenzi, Introduction to the theory of complexity, 2006, <https://www.pilucrescenzi.it/wp/wp-content/uploads/2017/09/itc.pdf>.
- [9] H. Broersma, F. Fomin, J. Nešetřil, G. Woeginger, More about subcolorings, *Computing* 69 (2002) 187–203.
- [10] J. Brown, The complexity of generalized graph colorings, *Discrete Appl. Math.* 69 (1996) 257–270.

- [11] M. Chudnovsky, N. Robertson, P. Seymour, R. Thomas, The strong perfect graph theorem, *Ann. Math.* 164 (2006) 51–229.
- [12] M. Chudnovsky, A. Scott, P. Seymour, S. Spirkl, Detecting an odd hole, *J. ACM* 67 (2020) 1–12, Article 5.
- [13] G. Cornuejols, X. Liu, K. Vušković, A polynomial algorithm for recognizing perfect graphs, in: *Proc. 44th IEEE Symposium on Foundations of Computer Science, FOCS2003*, 2003, pp. 20–27.
- [14] B. Courcelle, The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues, *RAIRO, Inform. Théorique et Appl.* 26 (1992) 257–286.
- [15] A. Čustiċ, B. Klinz, G. Woeginger, Geometric versions of the three-dimensional assignment problem under general norms, *Discrete Optim.* 18 (2015) 38–55.
- [16] R. Diestel, *Graph Theory*, fourth ed., in: *Graduate Texts in Mathematics*, vol. 173, Springer, 2012.
- [17] M. Dyer, L.A. Goldberg, C. Greenhill, M. Jerrum, The relative complexity of approximate counting problems, *Algorithmica* 38 (2004) 471–500.
- [18] M. Dyer, C. Greenhill, The complexity of counting graph homomorphisms, *Random Structures Algorithms* 17 (2000) 260–289.
- [19] M. Dyer, C. Greenhill, H. Müller, Counting independent sets in graphs with bounded bipartite pathwidth, *Random Structures Algorithms* 59 (2021) 204–237.
- [20] M. Dyer, M. Jerrum, H. Müller, K. Vušković, Counting weighted independent sets beyond the permanent, *SIAM J. Discrete Math.* 35 (2021) 1503–1524.
- [21] E. Eschen, C. Hoàng, M. Petrick, R. Sritharan, Disjoint clique cutsets in graphs without long holes, *J. Graph Theory* 48 (2005) 277–298.
- [22] E. Eschen, X. Wang, Algorithms for unipolar and generalized split graphs, *Discrete Appl. Math.* 162 (2014) 195–201.
- [23] A. Farrugia, Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard, *Electron. J. Comb.* 11 (2004) Research Paper #46.
- [24] T. Feder, P. Hell, Matrix partitions of perfect graphs, *Discrete Math.* 306 (2006) 2450–2460.
- [25] T. Feder, P. Hell, J. Stacho, G. Schell, Dichotomy for tree-structured matrix partition problems, *Discrete Appl. Math.* 159 (2011) 1217–1224.
- [26] M. Fellows, Parameterized complexity: the main ideas and connections to practical computing, *Electron. Notes Theor. Comput. Sci.* 61 (2002) 1–19.
- [27] J. Fiala, K. Jansen, V.B. Le, E. Seidel, Graph subcolorings: complexity and algorithms, *SIAM J. Discrete Math.* 16 (635–650) (2003).
- [28] F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. Combin. Theory Ser. B* 16 (1974) 47–56.
- [29] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, p. 1988.
- [30] C. Hoàng, On the complexity of finding a sun in a graph, *SIAM J. Discrete Math.* 23 (2010) 2156–2162.
- [31] M. Jerrum, Counting, Sampling and Integrating: Algorithms and Complexity, in: *Lectures in Mathematics – ETH Zürich*, Birkhäuser, 2003.
- [32] M. Jerrum, A. Sinclair, E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries, *J. ACM* 51 (2004) 671–697.
- [33] M. Kamiński, V. Lozin, M. Milaniċ, Recent developments on graphs of bounded clique-width, *Discrete Appl. Math.* 157 (2009) 2747–2761.
- [34] I. Kanj, C. Komusiewicz, M. Sorge, E. van Leeuwen, Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs, *J. Comput. System Sci.* 92 (2018) 22–47.
- [35] T. Kloks, *Treewidth – Computations and Approximations*, in: *Springer LNCS*, vol. 842, 1994.
- [36] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.* 2 (1972) 253–267.
- [37] G. MacGillivray, M.L. Yu, Generalized partitions of graphs, *Discrete Appl. Math.* 91 (1999) 143–153.
- [38] C. McDiarmid, N. YOLOV, Recognition of unipolar and generalised split graphs, *Algorithms* 8 (2016) 46–59.
- [39] D. Rose, R. Tarjan, G. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* 5 (1976) 266–283.
- [40] A. Salamon, P. Jeavons, Perfect constraints are tractable, in: *Proc. 14th International Conference on Principles and Practice of Constraint Programming (CP 2008)*, in: *Springer LNCS*, vol. 5202, 2008, pp. 524–528.
- [41] J. Stacho, *Complexity of Generalized Colourings of Chordal Graphs*, (Ph.D. thesis), Simon Fraser University, 2008, Chapter 5;
See also, J. Stacho, On 2-subcolourings of chordal graphs, in: *LATIN 2008: Theoretical Informatics*, in: *Springer LNCS*, 4957, 2008, pp. 544–554.
- [42] R. Tarjan, Decomposition by clique separators, *Discrete Math.* 55 (1985) 221–232.
- [43] R. Tyshkevich, A. Chernyak, Algorithms for the canonical decomposition of a graph and recognizing polarity, *Izv. Akad. Nauk. BSSR, Ser. Fiz. Mat. Nauk.* 6 (1985) 16–23, (in Russian).
- [44] L. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* 8 (1979) 189–201.
- [45] M. Yannakakis, Computing the minimum fill-in is NP-complete, *SIAM J. Alg. Disc. Meth.* 2 (1981) 77–79.