Thick Forests

Martin Dyer*
School of Computing
University of Leeds
Leeds LS2 9JT, UK
m.e.dyer@leeds.ac.uk

Haiko Müller*
School of Computing
University of Leeds
Leeds LS2 9JT, UK
h.muller@leeds.ac.uk

1 September 2023

Abstract

We consider classes of graphs, which we call thick graphs, that have their vertices replaced by cliques and their edges replaced by bipartite graphs. In particular, we consider the case of thick forests, which are a subclass of perfect graphs. We show that this class can be recognised in polynomial time, and examine the complexity of counting independent sets and colourings for graphs in the class. We consider some extensions of our results to thick graphs beyond thick forests.

1 Introduction

We consider classes of graph, which we call *thick graphs*, that can be represented with cliques as vertices and the connecting cobipartite graphs as edges. We call the graph given by shrinking the cliques to single vertices, the *thin graph*. The thin graph is usually far from unique. More formal definitions are given in section 2. We review previous work related to our definitions in section 1.1.

In particular, we are interested in whether a given graph has any thin graph lying in some particular graph class. However, this seems to be an NP-complete question for most graph classes, for example bipartite graphs. We examine this and related questions in section 2.

Thus we concentrate on a tractable case: thick forests. We show in section 3 that these are perfect graphs, resembling chordal graphs in many respects. Whereas chordal graphs are clique trees, thick trees are trees of cliques. However, thick forests are not chordal in general, and neither are all chordal graphs thick forests. Their intersection, chordal thick forests, has previously been considered by Stacho [49, 50], though with a different approach.

*Work supported by EPSRC grant EP/S016562/1 "Sampling in hereditary classes".

The union of these two classes, chordal graphs and thick forests, is properly contained in a larger class of perfect graphs which we call quasi thick forests. This class is characterised by clique cutset decomposition, which is our main tool throughout. We show that, like chordal graphs, thick forests can be recognised efficiently with the aid of clique cutset decomposition. The algorithm forms section 3.3 of the paper.

We then examine two classical graph problems on the class of thick forests, independent sets and q-colourings. That the decision version of these problems is in P follows directly from perfection, though we could give more efficient combinatorial algorithms for quasi thick forests. The counting versions are both #P-complete, however. We consider their complexity in a similar way to our work with Vušković [22] on claw-free perfect graphs, through clique cutset decomposition. Thus all our results apply equally to quasi thick forests. However, quasi thick forests are far from claw-free, since they contain all forests.

We show in section 4 that exactly counting all (weighted) independent sets in thick forests can be done in polynomial time. By contrast, we show in section 5 that exactly counting all q-colourings is #P-complete even in the case of cobipartite graphs (thick edges), but approximately counting all q-colourings has an FPRAS (fully polynomial randomised approximation scheme) for thick forests.

Finally, in section 6, we discuss some ways that our results can and cannot be extended, and some open questions.

1.1 Previous work

Thick graphs appear to have been introduced explicitly by Salamon and Jeavons [48] in the context of the Constraint Satisfaction Problem (CSP), as the complements of *microstructures* [35], particularly for perfect graphs. The satisfiability problem is then equivalent to finding the independence number of a thick graph with thin graph in some class. The class of thick trees was considered explicitly in [48]. Then the resulting tree-structured CSP can be solved efficiently.

However, the thick graph concept is implicit in earlier work on *subcolourings* of graphs, introduced by Albertson, Jamison, Hedetniemi and Locke [3]. Thus a q-subcolourable graph is a thick q-colourable graph and vice versa. The question here is to decide whether G has any thin graph H that is q-colourable. In particular, 2-subcolourability is the problem of testing for a thick bipartite graph [39]. Subcolourability has generated a literature of its own. See, for example, [27].

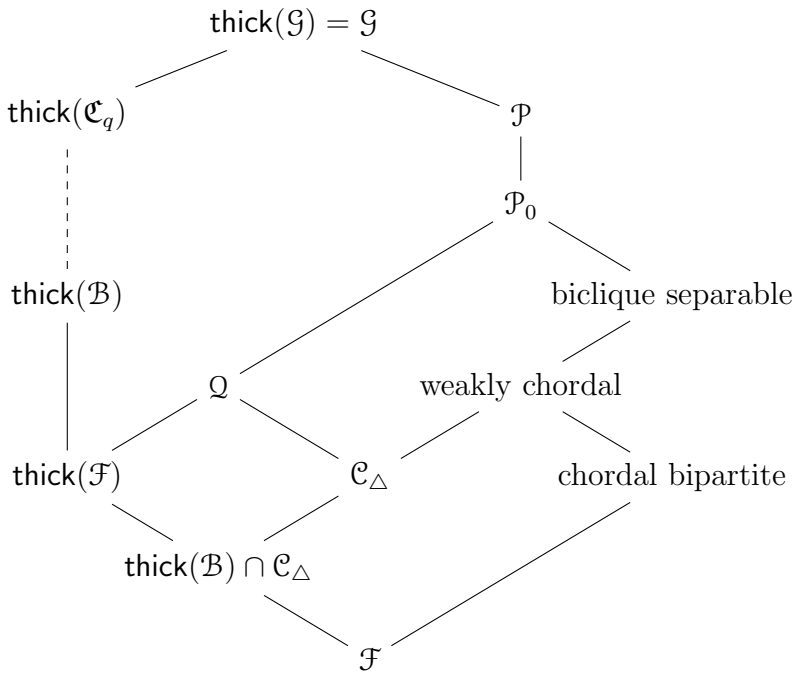
Thick trees have also been studied as *unipolar* graphs (e.g. [23, 24, 44]). In our teminology, these are thick *stars*. A star is a tree with one interior vertex, so these are a subclass of thick forests. Polynomial time algorithms for the recognition problem are given in [24], [44] and [52], and we give another in section 3.3.4 below. More recently, Kanj, Komusiewicz, Sorge and van Leeuwen [39] 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.

The case where the thin graph is a fixed graph H was called an (H, K)-partition of G by

MacGillivray and Yu [43]. They gave a complete solution to the recognition problem in this case. We will review their results in section 6.1. A related concept is that of *matrix partitions* [29], where we are given a fixed symmetric matrix M. If the matrix is the adjacency matrix of H but with 1's on the diagonal, 0's for the non-edges and $\star$'s for the edges, then a matrix partition is an (H, K)-partition.

We also show in section 3.2 that 2-subcolourable chordal graphs, as considered by Stacho [49, 50], are the chordal subclass of thick forests.

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



$\mathcal{P}$	perfect graphs	$\mathcal{F}$	forests
$\mathcal{P}_0$	(long hole)-free perfect graphs	$\mathcal{B}$	bipartite graphs ($\mathfrak{C}_2$)
$\mathcal{Q}$	quasi thick forests	$\mathcal{C}_\triangle$	chordal graphs
$\mathfrak{C}_q$	q-colourable graphs ($q > 2$)	$\mathcal{G}$	all graphs

Fig. 1: Graph classes and their inclusions

1.2 Preliminaries

$\mathbb{N}$ will denote the set of positive integers. We write $[m] = \{1, 2, \ldots, m\}$ 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 and undirected. A graph $G = (V(G), E(G))$ will have $n = |V|$ 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 having all possible edges, i.e. $E = V^{(2)} = \{vw : v, w \in V\}$. A *maximal* clique is not a proper subset 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 $vw \notin E$. 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.

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 clearly 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 $G[S]$. $G[S]$ is a *component* of G if 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 $\mathrm{N}(v)$, and $\mathrm{N}(v) \cup \{v\}$ by $\mathrm{N}[v]$. The neighbourhood $\mathrm{N}(S)$ of a set $S \subseteq V$ is the set $\bigcup_{v \in S} \mathrm{N}(v) \setminus S$, and $\mathrm{N}[S] = \bigcup_{v \in S} \mathrm{N}[v]$. We write $\deg(v)$ for the degree $|\mathrm{N}(v)|$ of v in G. We will also write $\bar{\mathrm{N}}(v)$ for $V \setminus \mathrm{N}(v)$, the set of non-neighbours of v, with similar extensions.

An edge set $F \subseteq E$ is a *cut* in G if, for some $\emptyset \neq V_1, V_2 \subseteq V$ with $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, we have $F = \{vw \in E : v \in V_1, w \in V_2\}$. We will write $F = V_1 : V_2$.

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 cutset*. A *maximal clique separator* (maximal clique separator) is a separator which is also a maximal clique. A *minimal clique separator* (maximal clique separator) is a minimal separator which is also a clique.

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 $\#\mathsf{ind}(G)$. 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, $e_1 \cap_2 = \emptyset$ for all $e_1, e_2 \in M$.

A *q-colouring* ψ of the vertices of a graph G, with colour set $Q = [q]$, is a function $\psi : V \to Q$. The colour classes are the sets $\psi^{-1}(i)$ for $i \in [q]$. It is a *proper* colouring if $\psi(v) \notin \psi(\mathrm{N}(v))$ for all $v \in V$, 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 $\#\mathsf{col}_q(G)$. The corresponding decision problem $\exists\mathsf{col}_q(G)$ asks: is $\#\mathsf{col}_q(G) > 0$? 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 $\chi(G) = 2$, and then its complement $\bar{G}$ is *cobipartite*, two cliques connected by the edges of a bipartite graph.

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 (k-)*antihole* is the complement $\bar{C}_k$ of a hole. Note that a 4-hole $\bar{C}_4 = 2K_2$ is disconnected, so antihole usually means the complement of a long hole. We will call this a *long antihole*. The *strong perfect graph theorem* (SPGT) [13] proves 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) [42], 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 [30], though only by an indirect optimisation algorithm. Exactly counting independent sets or q-colourings is #P-complete in general [21] and approximate counting is #BIS-hard [20], even for the class of bipartite graphs. Thus positive counting results seem 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 excluded. A hole of size 6 or more contains a $2K_2$, so its complement contains a 4-hole and is excluded. Thus chordal graphs are perfect. A chordal graph can be recognised by finding a *perfect elimination order* of its vertices. See [46], which gives a linear time algorithm for constructing the ordering.

A *chain graph* is a cobipartite chordal graph. It is the class of graphs which excludes triangles, 4-antiholes and 5-holes. Since it excludes $2K_2$, it excludes any hole or antihole of size greater than five, or path of more than three edges. It is a connected bipartite graph with nested complete bipartite neighbourhoods, plus an independent set of any size. See [34] for details. The complement of a chain graph will be called a *cochain* graph.

For graph theoretic background not given here, see Diestel [19], for example.

Finally, we use standard terms from computational complexity without comment. For further information see, for example, Bovet and Crescenzi [6] for a general introduction, Jerrum [37] for counting complexity and Fellows [26] for parameterised complexity.

2 Thick graphs

A *thick vertex* is a clique, and a *thick edge* is a cobipartite graph. Informally, a *thick graph* is a graph with thick vertices and thick edges.

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

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

is a *model* (H, ψ) of G. Intuitively, G is given by replacing each vertex in $\mathcal{V}$ by a thick vertex $\boldsymbol{v}$ and each edge in $\mathcal{E}$ by a thick edge $\boldsymbol{vw}$. We will say G is a *thick graph* of H and H is a *thin graph* of G, and write $G \in \mathsf{thick}(H)$ and $H \in \mathsf{thin}(G)$.

If $\boldsymbol{v} \in \mathcal{V}$ and $\psi^{-1}(\boldsymbol{v})$ is an r-clique $\{v_1, v_2, \ldots, v_r\}$ in G, we identify $\boldsymbol{v}$ with this clique, so $G[\boldsymbol{v}] \cong K_r$. Then, if $\boldsymbol{vw} \in \mathcal{E}$, the thick edge $\boldsymbol{vw}$ will be identified with the cobipartite graph $G[\boldsymbol{v} \cup \boldsymbol{w}]$, and we will say $\boldsymbol{w}$ is a thick neighbour of $\boldsymbol{v}$. We may omit the word "thick" for vertices, edges, neighbours and other constructs when it is implied by the context.

Thus, given a graph H, $\mathsf{thick}(H)$ is the class of graphs $\{G : G \text{ has a model } (H, \psi)\}$. Clearly $H \in \mathsf{thick}(H)$, by taking all cliques of size one. Similarly, given G, $\mathsf{thin}(G)$ is the class of graphs $\{H : G \text{ has a model } (H, \psi)\}$. Again, we have $G \in \mathsf{thin}(G)$ by taking all cliques of size one. More generally, any graph G can be regarded as a thick graph by choosing any *clique cover*, making its cliques the thick vertices and taking the thick edges as the edge sets between cliques.

Clearly $\mathsf{thin}(G)$ corresponds to the set of clique covers of G. For G an independent set, $|\mathsf{thin}(G)| = 1$. For G an n-clique, $|\mathsf{thin}(G)| = B_n$, the nth Bell number. For G triangle-free, $|\mathsf{thin}(G)|$ is the number of matchings in G, and thus even the size of $\mathsf{thin}(G)$ is a #P-Complete quantity in general [53].

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

Thus $G \in \mathsf{thick}(\mathcal{C})$ if there is any $H \in \mathcal{C}$ such that $G \in \mathsf{thick}(H)$. For given G and fixed $\mathcal{C}$, the problem of deciding if $G \in \mathsf{thick}(\mathcal{C})$ is the *recognition* problem for $\mathsf{thick}(\mathcal{C})$. Equivalently, we ask whether there is any $H \in \mathsf{thin}(G)$ such that $H \in \mathcal{C}$. In general, this problem is NP-complete. For example, if $\mathcal{C}$ is the class containing only the triangle, then the recognition for $G \in \mathsf{thick}(\mathcal{C})$ is the 3-colouring problem for $\bar{G}$. This is a case where $\mathcal{C} = \{H\}$ for a fixed graph H. More precisely, since we require it to be hereditary, $\mathcal{C}$ is the class of all induced subgraphs of H.

Some classes are related to natural graph parameters. For example, $\min\{|V(H)| : H \in \mathsf{thin}(G)\}$ is the *clique cover number* $\theta(G) = \chi(\bar{G})$. So asking for $\theta(G) \leq k$ is equivalent to $G \in \mathsf{thick}(\mathcal{G}_k)$, where $\mathcal{G}_k$ is the class of graphs with at most k vertices. Also $\min\{\chi(H) : H \in \mathsf{thin}(G)\}$ is the *subchromatic number* $\chi_{\mathrm{s}}(G)$ of G [3]. A graph is *q-subcolourable* if $\chi_{\mathrm{s}}(G) \leq q$. This is equivalent to $G \in \mathsf{thick}(\mathfrak{C}_q)$, where $\mathfrak{C}_q$ is the class of q-partite (q-colourable) graphs.

A graph G has $\chi_{\mathrm{s}}(G) = 1$ if and only if it is a *cluster graph*, or $G \in \mathsf{thick}(\mathcal{I})$, where $\mathcal{I}$ is the class of independent sets. Also $\chi_{\mathrm{s}}(G) = 2$ if and only if $G \in \mathcal{B}$, where $\mathcal{B} = \mathfrak{C}_2$ is the class of bipartite graphs. We will be interested in the class $\mathsf{thick}(\mathcal{F})$ of *thick forests*, where $\mathcal{F} \subset \mathcal{B}$ is the class of *forests*. We consider the recognition problem for thick forests in section 3.3, and counting independent sets and colourings in sections 4 and 5.

More generally, given two hereditary classes $\mathcal{C}_1, \mathcal{C}_2$ and any $G \in \mathcal{C}_1$, we ask whether there is any $H \in \mathsf{thin}(G)$ such that $H \in \mathcal{C}_2$. We call this class $\mathsf{thick}(\mathcal{C}_2|\mathcal{C}_1) = \mathsf{thick}(\mathcal{C}_2) \cap \mathcal{C}_1$. When $\mathcal{C}_1$ is the class $\mathcal{G}$ of all graphs, this is simply $\mathsf{thick}(\mathcal{C})$. When $\mathcal{C}_2$ is $\mathcal{G}$ this is simply $\mathcal{C}_1$, since the class of thick graphs which are in $\mathcal{C}_1$ is simply $\mathcal{C}_1$ since $G \in \mathsf{thick}(G)$.

2.1 Structure

Let ψ be any (not necessarily proper) q-colouring of $G = (V, E)$, that is, any function $\psi : V \to [q]$. Let $C_i = \psi^{-1}(i)$ be the colour classes. Let $\mathcal{F}$ be a set of graphs, with $F \in \mathcal{F}$ having a non-empty set of q-colourings $\Phi_F \subseteq \{\varphi : V(F) \to [q]\}$. Let $\Phi = \{(F, \varphi) : F \in \mathcal{F}, \varphi \in \Phi_F\}$, the set of forbidden coloured subgraphs. Clearly, F is a forbidden subgraph if $\Phi_F = \{\varphi : V(F) \to [q]\}$. Then we will say ψ is a Φ-free q-colouring of G if there is no induced subgraph F' of G such that $F' \cong F \in \mathcal{F}$ and $\psi_{|F'} \in \Phi_F$. Informally, $\psi(G)$ has no forbidden coloured subgraph.

Remark 1. This can be viewed as a CSP [16] on graphs, with all variables (vertices) having domain $[q]$. The set Φ then represents the constraints, though using the allowed values to give the *constraint language* $\Gamma = \{(F, \varphi) : F \in \mathcal{F}, \varphi \notin \Phi_F\}$. The above definition is intended to follow the forbidden subgraph characterisation of graph classes.

This includes *generalised colouring* [28]. Any hereditary graph class can be characterised by a set of excluded subgraphs. So let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q$ be hereditary classes, with forbidden subgraphs $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_q$. Let φ_i be the monochromatic colouring $\varphi_i(v) = i$ for all $v \in V(F)$ and $\Phi_F = \{\varphi_i : i \in [q]\}$ for all $i \in [q]$ and $F \in \mathcal{F}_i$. Then a Φ-free q-colouring is a generalised colouring, since ψ is a Φ-free q-colouring if and only if $G[C_i] \in \mathcal{C}_i$ for all $i \in [q]$. This is the definition of a generalised q-colouring. However, we will require the more general definition.

If $\mathcal{C}_i = \mathcal{C}$, and so $\mathcal{F}_i = \mathcal{F}$ for all $i \in [q]$, then a generalised q-colouring will be called an $\mathcal{F}$-free q-colouring. (This is called a P-colouring in [3], where P is our $\mathcal{C}$.) If $\mathcal{F} = \{F\}$ for some F, we will call it an F-free q-colouring. Thus a proper q-colouring of G is a K_2-free q-colouring.

For thick graphs, $\mathcal{C}$ is the class of cluster graphs for all $i \in [q]$. Then a generalised q-colouring is called a *q-subcolouring* [3], equivalently $G \in \mathsf{thick}(\mathfrak{C}_q)$. Thus $G \in \mathsf{thick}(\mathfrak{C}_q)$ if and only if it has a P_3-free q-colouring. So G has 1-subcolouring if it is a cluster graph and has a 2-subcolouring if it is a thick bipartite graph, so clearly a thick forest has $\chi_s(G) = 2$. Unfortunately, determining whether G has a q-subcolouring is NP-complete for any $q \geq 2$ [28].

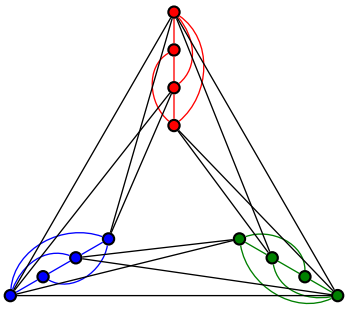


Fig. 2: A thick triangle with a 3-subcolouring

Lemma 1. *It is NP-complete to decide whether $G \in \mathsf{thick}(\mathfrak{C}_q)$ 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 [28] (see also [12]) 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 $\mathsf{thick}(\mathfrak{C}_{q-1})$. This remains valid if $q = \mathcal{O}(n^{1-\varepsilon})$, since polynomial in n^ε is equivalent to polynomial in n. □

Note that the additive condition in Lemma 1 is important, or recognition of unipolar graphs would be NP-complete, when it is not [24]. Also, Lemma 1 does not necessarily hold if G is restricted to lie in some graph class. For example, in [11] determining whether $G \in \mathsf{thick}(\mathfrak{C}_q|\mathcal{C})$ is in P if $\mathcal{C}$ is the class of interval graphs or the class of permutation graphs. We note that both these classes are subclasses of the class $\mathcal{P}$ of perfect graphs. However perfection is clearly not a sufficient condition since $\mathcal{B} \subset \mathcal{P}$.

However, the following shows that there are structural restrictions on graphs in $\mathsf{thick}(\mathcal{B})$. We have

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

From this we have

Lemma 3. *A thick bipartite graph G without long holes is perfect.*

Proof. We need only show that G contains no odd antihole. But $\bar{C}_5 = C_5$ excludes 5-antiholes, and Lemma 2 excludes larger odd antiholes. □

Let $\mathcal{P}_0$ denote the class of graphs without long holes or odd antiholes defined in [48]. Clearly $\mathcal{P}_0 \subset \mathcal{P}$, so is the class of perfect graphs without long holes. Thus we have

Corollary 1 (Salamon and Jeavons [48])**.** $\mathsf{thick}(F) \subseteq \mathcal{P}_0$.

Not all perfect graphs without long holes are thick forests or in the larger class $\mathcal{Q}$ of *quasi* thick forests defined in section 3.1 below. See, for example see Figs. 5 and 6. Thus the complexity

of recognising $G \in \mathsf{thick}(\mathcal{B}|\mathcal{P}_0)$ is an interesting question. This is not obviously in P, even for subclasses of $\mathcal{P}_0$, for example $\mathcal{F}$ and $\mathcal{C}_\triangle$, the class of chordal graphs. Stacho [49, 50] gave a polynomial time algorithm for recognition of $G \in \mathsf{thick}(\mathcal{B}|\mathcal{C}_\triangle)$. However, we show below, in Corollary 2, that $\mathsf{thick}(\mathcal{B}|\mathcal{C}_\triangle) = \mathsf{thick}(\mathcal{F}|\mathcal{C}_\triangle)$. So recognition for $\mathsf{thick}(\mathcal{B}|\mathcal{C}_\triangle)$ is implied by recognition for $\mathsf{thick}(\mathcal{F})$, which we will show in section 3.3.

Eschen, Hoàng, Petrick and Sritharan [25] defined another subclass of $\mathcal{P}_0$ which they called *biclique separable* graphs. This is the hereditary class graphs which are either cliques, or have a separator consisting of two disconnected cliques, and are also long hole-free. This class strictly includes *weakly chordal* graphs, but does not include $\mathsf{thick}(\mathcal{F})$ or $\mathcal{Q}$, since all atoms in the decomposition of [25] must be cliques, and there are cobipartite graphs which are not biclique separable. The graph on the left in Fig. 3 is an example. This graph is cobipartite, not a clique and has no clique cutset. The only disconnected 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 are all biclique separable graphs in $\mathcal{Q}$. The graph on the right in Fig. 3 is an example. It is biclique separable, in fact weakly chordal, but is not a clique, not cobipartite and has no clique cutset. So it is not in $\mathcal{Q}$.

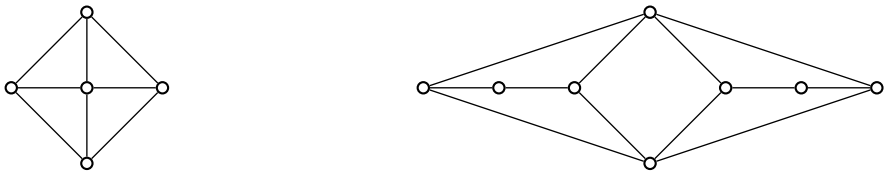


Fig. 3: Biclique separable is incomparable with $\mathcal{Q}$

Graphs in the class $\mathcal{P}_0$ clearly have much structure, but this structure has not been investigated in general. Recognition is clearly in P, using the algorithms of [17] and [14] on $\bar{G}$. But this gives little insight to algorithmic questions on $\mathcal{P}_0$ rather than $\mathcal{P}$. 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 $G \in \mathcal{Q}$, but it is #P-complete for $G \in \mathcal{P}$.

3 Thick forests

3.1 Structure

Thick forests have more structure than general thick graphs, since they have a *clique cutset decomposition*. See [5] for definitions and some recent applications, and Tarjan [51] for an efficient algorithm. A thick tree G has a clique cutset corresponding to every internal vertex of the thin tree H, since every internal vertex is a separator for H. Using Tarjan's algorithm, the decomposition of any graph G can be done in $\mathcal{O}(mn)$ time. A leaf of the decomposition tree (called an *atom*) for a thick tree must either a clique or a cobipartite graph. These can be recognised in $\mathcal{O}(m)$ time, as we show below. However, a clique cutset decomposition does not necessarily recover the tree structure, for two reasons. Firstly, G may have many

clique cutsets which are not thick vertices. Secondly, the clique cutsets in a clique cutset decomposition are not disjoint in general, as they should be in a thick tree representation.

We will use clique cutset decomposition to count colourings and independent sets in G in sections 4 and 5. In fact, the class of graphs for which these algorithms are effective is larger than thick forests. We can define a graph class by requiring that all the atoms of any clique cutset decomposition are either cliques or cobipartite graphs. We call this the class $\mathcal{Q}$ of *quasi* thick forests. A graph in $\mathcal{Q}$ can clearly be recognised in $\mathcal{O}(mn)$ time using Tarjan's algorithm [51]. Quasi thick forests include the class $\mathcal{C}_\triangle$ of chordal graphs, where the atoms of the clique cutset decomposition are all cliques. (A graph has clique cutset decompositions with cliques as atoms if and only if it is chordal [31].) A chordal thick tree $G \in \mathcal{F} \cap \mathcal{C}_\triangle$ is shown in Fig. 4.

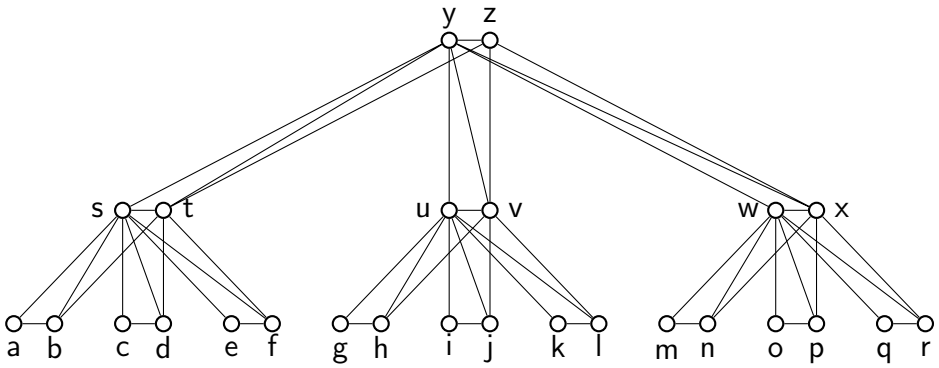


Fig. 4: A chordal thick tree with K_2 vertices

Clearly not all thick forests are chordal, but neither are all chordal graphs thick forests. The graph shown in Fig. 5 is an example of a chordal graph which is not a thick tree. We will prove this in section 3.3.2.

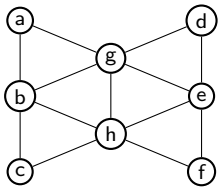


Fig. 5: A chordal graph which is not a thick tree

We may also observe that a non-chordal quasi thick tree is not necessarily a thick tree. See Fig. 6 for an example, but we defer proving this to section 3.2. Quasi thick forests and chordal graphs are clearly polynomial time recognisable. But quasi thick forests are not simply the union of thick forests and chordal graphs, so the recognition of thick forests $\mathcal{F}$ is not obviously in polynomial time. We will consider the problem of polynomial time recognition of $\mathcal{F}$ in section 3.3. We will show as examples that the graph of Fig. 4 is a thick tree, and the graph of Fig. 5 is not. However, first we will use clique cutset decomposition to establish some properties of quasi thick forests $\mathcal{Q}$, which imply the same properties for $\mathcal{F}$.

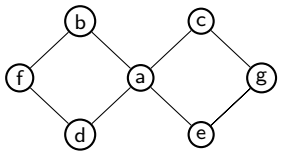


Fig. 6: A quasi thick tree which is not a thick tree

Lemma 4. *Let H be a graph with no clique cutset. Then a graph G can contain an induced copy of H only in the atoms of its clique cutset decomposition.*

Proof. If H properly intersects two atoms, it has a clique cutset. □

We will use this via the following.

Lemma 5. *A hole or (long) antihole has no clique cutset.*

Proof. It is clear that a hole has no clique cutset. So let the antihole be $\bar{C}$, where C is a long hole. Any vertex in $\bar{C}$ has only two non-neighbours, so any clique cutset S must separate one vertex v from at most two others. Thus $|S| \geq 2$ and we must have $\mathrm{N}(v) \subseteq S$. But $\mathrm{N}(v)$ induces a path in C, so S has at least one non-edge in $\bar{C}$, a contradiction. □

The following is then a directl consequence o the SPGT and Lemmas 4 and 5.

Lemma 6. *A graph is perfect if and only if all atoms of its clique cutset decomposition are perfect.*

The class $\mathcal{Q}$ has stronger properties. We know that $\mathcal{F} \subset \mathcal{Q}$. Now recall that $\mathcal{P}_0$ is the class of long hole-free perfect graphs.

Lemma 7. $\mathcal{Q} \subset \mathcal{P}_0$.

Proof. The atoms of the clique cutset decomposition for $G \in \mathcal{Q}$ are either cliques or cobipartite graphs. Neither can contain a long hole from Lemmas 4 and 5. A clique contains no hole or antihole. A cobipartite graph contains no long hole and is perfect, so contains no odd antihole. Perfection follows directly from the SPGT. Thus $\mathcal{Q} \subseteq \mathcal{P}_0$.

To see that $\mathcal{Q} \neq \mathcal{P}_0$, observe that otherwise any atom in the clique cutset decomposition of $G \in \mathcal{P}_0$ must be a clique or cobipartite. Thus we need only exhibit possible atoms which are neither. But any chordal bipartite graph G with no cut edge could be an atom, yet is not a clique or cobipartite. It has no long holes because it is chordal bipartite. It is perfect since it is bipartite, and so has no odd antiholes. It has no clique cutset, since its maximal cliques are edges, and it has no cut edge by assumption.

Complete bipartite graphs are chordal bipartite graph with no cut edge. Thus the first of the excluded subgraphs for thick forests in Fig. 9 below is also a forbidden subgraph for $\mathcal{Q}$, since it is $K_{2,3}$. However, none of the other graphs in Figs. 9 and 10 are forbidden for $\mathcal{Q}$, since either they are chordal or they have cut edges or both. □

This clearly implies Corollary 1 above and strengthens the result $\mathcal{F} \subseteq \mathcal{P}_0$ given for thick forests in [48]. This is also given for unipolar graphs in [24], although these are a subclass of thick forests.

Cobipartite graphs can contain even antiholes. In fact, an even antihole $\bar{C}_{2k}$ is a connected cobipartite graph for $k \geq 3$. The class of (long hole, odd antihole)-free graphs has not received much attention. A subclass of the complementary class (odd hole, long antihole)-free is investigated in [41]. Note that these classes can be recognised in polynomial time using the algorithms in [14, 45], for example. However, a thick edge, a cobipartite graph can be recognised more easily.

Lemma 8. *A cobipartite graph, $G = (V_1 \cup V_2, E)$ is recognisable in $\Theta(n^2)$ time.*

Proof. Let $n_1 = |V_1|$, $n_2 = |V_2|$, so $n = n_1 + n_2$ and $|E| \leq n_1 n_2$. Now

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

where we have used the convexity of $x(x-1)$ on the left and $n = n_1 + n_2$ on the right. So $m = \Theta(n^2)$, and so any algorithm recognising G must be $\Omega(n^2)$.

We determine the connected components $C_1, \ldots, C_k$ of the bipartite graph $\bar{G} = (V_1 \cup V_2, \bar{E})$ in $\mathcal{O}(n^2)$ time. These must all be bipartite or G is not cobipartite. From this we can determine V_1 and V_2 in 2^k ways, by assigning vertices of the bipartition of the C_i to V_1 and V_2. Then $G(V_1 \cup V_2)$ is a cobipartite graph. The overall time to compute any one of these 2^k graphs is clearly $\mathcal{O}(n^2)$. □

If G is chordal, a thick edge must be a cobipartite chordal graph. For these we have

Lemma 9. *A cobipartite graph is chordal if and only if it is a cochain graph.*

Proof. By definition, the complement is a bipartite graph and must exclude $\bar{C}_4 = 2K_2$. But these two properties define chain graphs. See [54] for further information. □

The clique cutset decomposition returned by the algorithm of [51] does not necessarily reveal the thick tree structure of the graph, even if it has one. So the complexity of the thick forest recognition problem is not clear. However, we now give some properties of classes of thick graphs which we use to construct a recognition algorithm for thick forests in section 3.3.

If $\boldsymbol{u}$ and $\boldsymbol{w}$ are (thick) vertices of a thick graph G, we will say that a vertex $u \in \boldsymbol{u}$ is *movable* to $\boldsymbol{w}$ if G is also a thick graph with vertices $\boldsymbol{u} \setminus u$, $\boldsymbol{w} \cup u$, and all other vertices unaltered. Moving vertices can be used to make local changes to the model (H, ψ) of G, but does not alter G. In particular it does not affect the connectivity properties of G.

Lemma 10. *A vertex $u \in \boldsymbol{u}$ is movable between $\boldsymbol{u}$ and $\boldsymbol{w}$ if and only if u is complete to $\boldsymbol{w}$.*

Proof. $\boldsymbol{w} \cup u$ is a clique if and only if u is complete to $\boldsymbol{w}$. Thus, in particular, $\boldsymbol{w}$ must be a neighbour of $\boldsymbol{u}$. Conversely, if $\boldsymbol{w} \cup u$ is a clique then since u is complete to $\boldsymbol{u} \setminus u$, it can be moved back to $\boldsymbol{u}$, as we should expect. □

Note that, if we wish to move a vertex between thick vertices, we must be able to identify both thick vertices, in order to check that it is complete to both.

Not all vertices that are movable can be moved in G without changing $\mathsf{thin}(G)$. We will call a vertex *mobile* if it can be moved while preserving $\mathsf{thin}(G)$. Let $\mathcal{T}$ be the class of triangle-free graphs.

Lemma 11. *If $G \in \mathsf{thick}(\mathcal{T})$ then $v \in \boldsymbol{u}$ is mobile if and only if $\{v\} \subset \boldsymbol{u}$ and $\mathrm{N}(v) \subseteq \boldsymbol{w}$ for some neighbour $\boldsymbol{w}$ of $\boldsymbol{u}$. If $G \in \mathcal{F}$, we can remove the condition $\{v\} \subset \boldsymbol{u}$.*

Proof. If $\{v\} \subset \boldsymbol{u}$ and $\mathrm{N}(v) \subseteq \boldsymbol{w}$ then moving v to $\boldsymbol{w}$ does not change $\mathsf{thin}(G)$. If $\{v\} = \boldsymbol{u}$, then moving v to $\boldsymbol{w}$ removes the edge $\boldsymbol{uw}$ and so changes $\mathsf{thin}(G)$. If $\{v\} \subset \boldsymbol{u}$, $\mathrm{N}(v) \cap \boldsymbol{w} \neq \emptyset$ and $\mathrm{N}(v) \cap \boldsymbol{w'} \neq \emptyset$ for $\boldsymbol{w} \neq \boldsymbol{w'}$, then moving v to $\boldsymbol{w}$ creates a thick triangle $\boldsymbol{uww'}$. The modified thick graph $G' \notin \mathsf{thick}(\mathcal{T})$, so $\mathsf{thin}(G)$ has been changed. If $\{v\} = \boldsymbol{u}$, we can move v to $\boldsymbol{w}$, contracting the edge $\boldsymbol{uw}$, and giving a different graph in $\mathsf{thin}(G) \cap \mathcal{F}$. □

Thus we have shown that, for a graph $G \in \mathsf{thick}(\mathcal{T})$, two different representations of G can only be obtained from each other by moving mobile vertices along thick edges. This greatly restricts the freedom to make such changes, and permits recognition for $\mathcal{F}$.

We will now characterise maximal clique separators for graphs in $\mathsf{thick}(\mathcal{T})$, which we will use as the basis for recognition of $G \in \mathcal{F}$.

Lemma 12. *Let $G \in \mathsf{thick}(\mathcal{T})$. Then, for any maximal clique separator A in G, one of the following applies,*

(a) $\boldsymbol{u} = A$, for a thick vertex $\boldsymbol{u}$,

(b) $\boldsymbol{u} \subset A$, for a unique thick edge $A \subseteq \boldsymbol{uw}$,

(c) $\boldsymbol{u} \setminus A, \boldsymbol{w} \setminus A \neq \emptyset$, for a unique thick edge $A \subset \boldsymbol{uw}$.

If $G \in \mathcal{F}$, then we may additionally assume that $A \subset \boldsymbol{uw}$ in (b).

Proof. Suppose $G \in \mathsf{thick}(H)$, where $H \in \mathcal{T}$. A thick vertex $\boldsymbol{u}$ is a maximal clique separator A if no vertex in any of its neighbours is complete to A. Otherwise, A can only intersect two vertices $\boldsymbol{u}$ and $\boldsymbol{w}$. If it were to intersect three, $\boldsymbol{u}$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, at least two must be non-adjacent, since $H \in \mathcal{T}$, and hence vertices from all three cannot lie in the same clique. So A is contained in the thick edge $\boldsymbol{uw}$. Thus, if we have $\boldsymbol{u} \subset A$, for a thick vertex $\boldsymbol{u}$, A must include $U \subset \boldsymbol{u}$ and $W \subset \boldsymbol{w}$, connected by a complete bipartite graph with bipartition U, W. Thus $A = U \cup W$ is properly contained in the thick edge $\boldsymbol{uw}$.

If $G \in \mathcal{F}$ and $A = \boldsymbol{u} \cup \boldsymbol{w}$, we can contract $\boldsymbol{uw}$, giving a tree $H' \in \mathsf{thin}(G)$. □

Remark 2. The three cases in Lemma 12 form the basis of the recognition of thick forests in section 3.3. We will refer to them as maximal clique separators of types (a), (b) or (c), respectively.

Remark 3. A thick vertex is a separator unless it is a leaf, but we cannot assume that it is a maximal clique separator. See Fig. 4 for example, where all the vertices are K_2's, but all the maximal clique separators are K_3's.

Remark 4. The edge contraction used in strengthening Lemmas 11 and 12 from $\mathcal{T}$ to $\mathcal{F}$ is not valid more generally. The only subclass of $\mathcal{T}$ which is closed under contraction is $\mathcal{F}$. Any cycle in a graph $H \in \mathcal{T} \setminus \mathcal{F}$ can be contracted to a triangle, so the resulting graph $H' \notin \mathcal{T}$.

Remark 5. All vertices movable between $\boldsymbol{u}$ and $\boldsymbol{w}$ will be in any maximal clique separator $A \subseteq \boldsymbol{uw}$. They are complete to $\boldsymbol{u}$ and $\boldsymbol{w}$ and hence to A, so they must be in A because it is a maximal clique. These vertices might or might not be mobile.

Recognition of a thick vertex is clearly in $\Theta(n^2)$ time, since $m = n(n-1)/2$. Recognition of a thick edge has the same time bound, as shown in Lemma 8 above.

Note that if a thick tree has any chordal thick edge, the clique cutset decomposition returned by the algorithm of [51] may not reveal its thick tree structure. So the complexity of the thick forest recognition problem is not obvious even for chordal graphs. However, we will now give some properties of classes of thick graphs which we will use to construct a recognition algorithm for thick forests in section 3.3.

We now show that a thick edge can contain only a linear number of "internal" maximal clique separators.

Lemma 13. *At most* $\min\{|\boldsymbol{u}|, |\boldsymbol{w}|\} - 1$ *maximal clique separators can be properly contained in a thick edge* $\boldsymbol{uw}$.

Proof. Assume without loss of generality that $|\boldsymbol{u}| \leq |\boldsymbol{w}|$. Let A be any maximal clique separator in $\boldsymbol{uw}$, with $U = \boldsymbol{u} \cap A \neq \emptyset$, $W = \boldsymbol{w} \cap A \neq \emptyset$ and $A = U \uplus W$. Since A is a separator, any edge from $\boldsymbol{w}$ must meet U. Suppose $A' = U' \uplus W'$ is another maximal clique separator. If $W' \neq W$, then any edge from $W' \setminus W$ meets $U' \cap U$, so either $W' \subseteq W$ or $U' \subseteq U$. If $U' = U$, then $A'' = U \uplus (W \cup W')$ is a larger clique separator, contradicting the maximality of A, and similarly if $W' = W$. Thus either $U' \subset U$, $W' \supset W$ or $U' \supset U$, $W' \subset W$. In either case $|U'| \neq |U|$. Thus $A = U \uplus W$ is the unique maximal clique separator with $|A \cap \boldsymbol{u}| = |U|$. Since there are only $|\boldsymbol{u}| - 1$ possible values of $|U|$, the result follows. Note that the cases $U = \boldsymbol{u}$ and $W = \boldsymbol{w}$ are not included. □

Remark 6. Though the cases $U = \boldsymbol{u}$ and $W = \boldsymbol{w}$ are not included in this count, they can be maximal clique separators.

Lemma 14. *The bound in Lemma 13 is tight in the worst case.*

Proof. For an integer $k \geq 2$, let $\boldsymbol{u} = \{u_i : i \in [k]\}$, $\boldsymbol{w} = \{w_j : j \in [k]\}$ and $\boldsymbol{uw} = \{u_i w_j : i \geq j\}$. Then, for any $r \in [k-1]$ the vertices $U_r = \{u_i : i \geq r\}$ and $W_r = \{v_j : j \leq r\}$ induce a maximal clique separator $A_r = U_r \cup W_r$. This is a clique since U_r, W_r are cliques and $U_r \cup W_r$ induces a complete bipartite graph, because every vertex in U_r is connected to every vertex in W_r. It is a separator because no vertex in $\bar{U}_r = \{u_i : i < r\}$ is adjacent to any vertex in $\bar{W}_r = \{w_j : j > r\}$. It is maximal since no vertex in $\bar{U}_r$ is adjacent to w_r and no vertex in $\bar{W}_r$ is adjacent to u_r. There are $k - 1 = \min\{|\boldsymbol{u}|, |\boldsymbol{w}|\} - 1$ such cliques. □

Remark 7. Note that the graph constructed in Lemma 14 are cochain graphs, which are chordal. Since any thick edge can be triangulated to give a chordal graph by adding edges, the bound of Lemma 14 is only attained for chordal thick edges.

Using this we can determine the maximal clique separators in a cobipartite graph.

Lemma 15. *If $G = (\boldsymbol{u} \cup \boldsymbol{w}, E)$ is a cobipartite graph, then its at most $\lfloor n/2 \rfloor - 1$ maximal clique separators can be listed in $\mathcal{O}(mn) = \mathcal{O}(n^3)$ time.*

Proof. We use the method of Lemma 8 to determine $\boldsymbol{u}$, $\boldsymbol{w}$ and $E' = \{uw : u \in \boldsymbol{w}, w \in \boldsymbol{w}\}$ in $\mathcal{O}(n^2)$ time.

We use the algorithm of [46] to triangulate G, giving a chordal graph $G^* = (\boldsymbol{u} \cup \boldsymbol{w}, E \cup F)$, where F is the set of added edges. All edges in F of the form uw with $u \in \boldsymbol{u}$, $w \in \boldsymbol{w}$. This requires $\mathcal{O}(mn) = \mathcal{O}(n^3)$ time. (In fact, this can be done in $\mathcal{O}(n^{2.376})$ time, see [33].) From Lemma 9, G^* must be a cochain graph.

Now we use the algorithm of [4] to determine the maximal cliques $C_1, C_2, \ldots, C_k$ in G^* in $O(m)$ time, where $k < n$. (This can be simplified for a cochain graph, but we will not do so.) These can then be listed in $\mathcal{O}(mn) = \mathcal{O}(n^3)$ time. The space requirement is only $O(n)$ since the cliques can be generated in the order they are needed.

Observe that a maximal clique C_i in G^* comprises $U_i = C_i \cap \boldsymbol{u}$ and $W_i = C_i \cap \boldsymbol{w}$, where $W_i = \{v \in \boldsymbol{w} : v \text{ is complete to } U_i\}$. So C_i is uniquely determined by U_i. This remains true for any cobipartite graph, in particular G.

We now inspect the list and determine the sublist of those that are separators in G^* in $O(mn)$ time. These are maximal cliques C_i such that there is any $uw \in E' \cup F$ with $u \notin U_i$, $w \notin W_i$. Let us assume these are renumbered $C_1, C_2, \ldots, C_\ell$. These are separators in G^* and also separators in G, since deleting edges cannot change a non-separator into a separator. But they may not be cliques.

We inspect each separator C_i in the list. We determine $X_i = \{w \in W_i : \exists u \in U_i, uw \in F\}$. In G, no vertex in X_i is complete to U_i, but every vertex in $W_i' = W_i \setminus X_i$ is complete to U_i. Thus $C_i' = U_i \cup W_i'$ is a clique in G. It is also maximal, since otherwise C_i would not have been maximal in G'. Thus C_i' is a maximal clique separator and, from Lemma 13, C_i' is the unique maximal clique separator with U_i of size $|U_i|$.

Thus we can list the cliques $C_1', C_2', \ldots, C_\ell'$ in time $\mathcal{O}(mn) = \mathcal{O}(n^3)$ time, from Lemma 8. Also $\ell \leq n_1 - 1 \leq \lfloor n/2 \rfloor - 1$, assuming $|\boldsymbol{u}| = n_1 \leq n_2 = |\boldsymbol{w}|$, by Lemma 13. □

Clearly there are at most $(n-2)$ maximal clique separators of type (a) in a thick forest, since a forest must have at least two leaves. This bound achieved by a path with all cliques of size one. It also follows from Lemma 13 that the number of maximal clique separators properly contained in thick edges is $\mathcal{O}(n)$. But, by contrast, there can be an exponential number of maximal clique separators of types (b) or (c). This is because these maximal clique separators are simply minimal clique separators expanded to maximal cliques, and the expansion may be possible in many ways. See Tarjan [51].

Lemma 16. *A thick forest $G = (V, E)$ on n vertices can have at least $2^{\lfloor n/2 \rfloor - 1}$ maximal clique separators of type (b).*

Proof. Consider the graph C, a clique on $2k$ vertices with vertex set $V = \boldsymbol{u} \cup \boldsymbol{w}$, where $\boldsymbol{u} = \{u_i : i \in [k]\}$ and $\boldsymbol{w} = \{w_i : i \in [k]\}$. Let $G_0 = C \setminus M$, where $M = \{u_i w_i : i \in [k]\}$

is a matching of size k. Now G_0 is a cobipartite graph since M is bipartite. Attach u_1 to a new vertex u' and w_1 to a new vertex w' to give $G = G_0 \cup \{u', w'\}$, with $n = 2k + 2$ vertices. (See Fig. 7.) Thus G is a thick P_4 with thick vertices $\{u'\}, \boldsymbol{u}, \boldsymbol{w}, \{w'\}$. Each of the 2^k sets $X = \{x_i : x_i \in \{u_i, w_i\}, i \in [k]\}$ induces a k-clique in G_0 which separates u' from w', since it contains either u_1 or w_1. Moreover, each such X is a maximal clique, since every vertex $V \setminus X$ has a non-edge $u_i v_i$ to X. Thus there are at least 2^k maximal clique separators X in G, and they are all of type (b), since $X, V \setminus X$ is a cobipartition of G_0. □

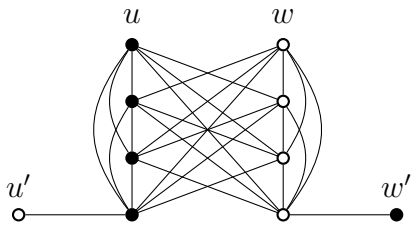


Fig. 7: Graph of Lemma 16 with $k = 4$

Lemma 17. *A thick forest $G = (V, E)$ on n vertices can have more than $(1.74)^{\lfloor n/2 \rfloor - 3}$ maximal clique separators of type (c), for all $n > 10$.*

Proof. Consider the graph G_0 given by the intersection of two $(2k + 1)$-cliques U, W in a $2k$-clique C, with. $U \setminus C = u_0$, $W \setminus C = w_0$. Let $G_1 = G_0 \setminus H$ where H is a is a $2k$-cycle through all the vertices of C. Now $C \setminus H$ is a cobipartite graph $\boldsymbol{u'w'}$ with unique bipartition $\boldsymbol{u', w'}$ of size k. Let $\boldsymbol{u'} = \{u_i : i \in [k]\}$, $\boldsymbol{w'} = \{w_i : i \in [k]\}$. Then G_1 is a cobipartite graph with exactly two possible bipartitions $\boldsymbol{u'} \cup u_0, \boldsymbol{w'} \cup w_0$ or $\boldsymbol{u'} \cup w_0, \boldsymbol{w'} \cup u_0$. Attach u_0 to a new vertex u' and w_0 to a new vertex w' to give $G = G_0 \cup \{u', w'\}$. So G has $n = 2k + 4$ vertices, i.e. $k = n/2 - 2$, and is a thick P_4. (See Fig. 8, where $H = (u_1 w_2 u_2 w_3 u_3 w_4 u_4)$.) Let X be any maximal independent set in $\bar{G}_1$, so X is a maximal clique in G_1. Then X comprises a maximal independent set in H plus one of u_0, w_0, but not both. Thus the number of such maximal cliques in G_1 is twice the number of maximal independent sets in a $2k$-cycle, which is $2E_{2k}$, where E_i is the ith *Perrin number* [47]. Each such X is a separator since it contains either u_0 or w_0. So G contains $2E_{2k}$ maximal clique separators. None are of type (a). Only four are of type (b), corresponding to a choice of *maximum* independent set $\boldsymbol{u'}$ or $\boldsymbol{w'}$ in H, and a choice of u_0 or w_0. Thus G contains $2(E_{2k} - 2)$ maximal clique separators of type (c). From [47], we have $E_{2k} \geq (1.32)^{2k-2} > (1.74)^{k-1}$ for all $k > 1$. Now $2(E_{2k-2} - 2) \geq E_{2k-2}$ if $E_{2k-2} \geq 4$, which is true if $k > 3$, giving $n > 10$. □

Remark 8. Note that all the maximal clique separators in Lemma 16 are simply expansions into the thick edge $\boldsymbol{uw}$ of the (unique) minimal clique separators $\{u_1\}$ and $\{w_1\}$. Similarly all those in Lemma 17 are simply expansions of $\{u_0\}$ and $\{w_0\}$.

Since all maximal clique separators expanded into the same thick edge are equivalent for identifying structure, these exponential numbers are not problematic. They only preclude algorithms which list all maximal clique separators, as is done with maximal cliques in algorithms for chordal graphs. See, for example, [23] for chordal unipolar graphs. This is

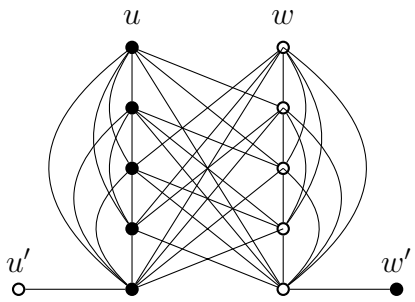


Fig. 8: Graph of Lemma 17 with $k = 4$

unfortunate, since maximal clique separators in thick trees play a similar role to maximal cliques in chordal graphs in most respects. But, in spite of this apparent difficulty, we base our algorithm for recognising thick forests in section 3.3 on maximal clique separators of types (a) and (b), and only a special case of type (c). That case is where the maximal clique separator lies in a thick edge $\boldsymbol{uw}$ and $\boldsymbol{u}$ is a leaf.

3.2 Excluded subgraphs

Thick forests can be characterised by minimal excluded subgraphs, but there appear to be too many to be useful. Some examples are given in Fig. 9, but we will not prove these, except for the case of the graph of Fig. 5 that appears in Fig. 9. We defer proof to section 3.3.2 below. The others can be proved similarly. There are even infinite families of minimal excluded subgraphs besides long holes and odd antiholes. One such family is shown in Fig. 10. We will not prove this, but it can be shown not to be a thick tree using the algorithm of section 3.3, 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.

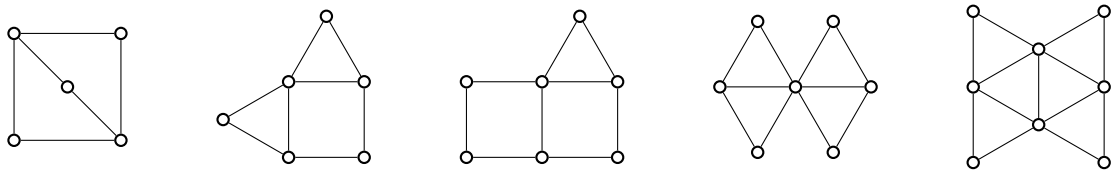


Fig. 9: Some minimal forbidden induced subgraphs for thick forests

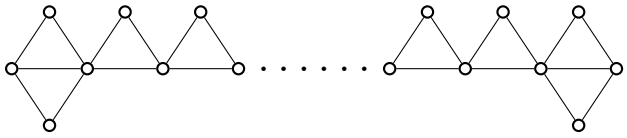


Fig. 10: An infinite family of minimal forbidden subgraphs for thick forests

A simpler description can be given using the excluded coloured subgraph approach of section 2.1. We will now state and prove this characterisation.

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. 11.)

Fig. 11: Alternating and cobipartite colourings of a 4-hole.

Proof. Suppose G is a thick forest with thin forest H. Then it is a thick bipartite graph, so must have a 2-colouring satisfying (b). It must satisfy (a) by Lemma 7. Also, by Lemma 4, any 4-hole must lie in a thick edge, so it must have the cobipartite colouring (see Fig. 11), 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 thick edge and an edge of G in a thick vertex, 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 thick edge since its monochromatic edges must lie in two different thick vertices and its dichromatic edges must lie in the bipartite graph connecting them. Thus H can have no triangles or holes and must be a forest. □

Corollary 2. *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. □

Thus the graph class considered in [49, 50] is precisely the class of chordal thick forests. In our notation, we have shown that $\mathsf{thick}(\mathcal{B}|\mathcal{C}_\triangle) = \mathsf{thick}(\mathcal{F}|\mathcal{C}_\triangle)$.

These forbidden coloured subgraph characterisations do not immediately imply polynomial time recognition algorithms. Recognising a thick bipartite graph, that is, a graph satisfying only (b), is NP-complete from Lemma 1. On the other hand, pure forbidden subgraph characterisations do not necessarily imply polynomial time recognition when there are infinitely many forbidden subgraphs, as there are here.

As an application of Theorem 1 consider the graph of Fig. 6, two C_4's $\{\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{d}\}$and $\{\mathsf{a}, \mathsf{c}, \mathsf{g}, \mathsf{e}\}$ with one common vertex a. This is a quasi thick tree since decomposing on the clique $\{\mathsf{a}\}$ gives the two C_4's as atoms, and both are cobipartite. However, there are four P_3's through a, and so any P_3-free 2-colouring must use the alternating colouring at least on one of the

C_4's. So this non-chordal graph is a quasi thick tree but is not a thick tree, as claimed in section 3.1.

3.3 Recognition

We have seen in Lemma 1 that it is NP-complete to recognise thick bipartite graphs in general, given only their graph structure. However, this intractability does not necessarily apply to subclasses of thick bipartite graphs. Thus, we will now show that thick forests can be recognised in polynomial time.

Remark 9. Note that we cannot guarantee to find any particular representation of G as a thick tree, since $\mathsf{thin}(G)$ may be exponentially large, as discussed in section 2. Rather, we construct an $H \in \mathsf{thin}(G) \cap \mathcal{F}$ if there is one, and hence failure will imply that $\mathsf{thin}(G) \cap \mathcal{F} = \emptyset$. Thus, in what follows, thick vertices $\boldsymbol{u}, \boldsymbol{w}, \ldots$ should not be thought of as fixed in advance, but rather as vertices of $H \in \mathcal{F}$ in some solution to $H = \mathsf{thin}(G)$, if this equation is satisfiable.

Stacho [49, 50] showed that there is a polynomial time algorithm to deciding whether a chordal graph is 2-subcolourable. From Corollary 2, we know that this is the same problem as recognising that a chordal graph is a thick forest. Thus, for example, Stacho's algorithm should verify that the graph of Fig. 4 is a thick tree. However, Stacho's algorithm is restricted to chordal thick trees, and does not appear to give the thick tree representation. So we will give a different algorithm here, which recognises a general thick forest. Note also, from Lemma 14, that chordal thick edges have the maximum number of maximal clique separators, so may be the most difficult to recognise, rather than the easiest.

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 or edge. Thus, if we can identify any thick vertex or thick edge of G, we can decompose the recognition problem in a divide-and-conquer fashion.

Note also that, for any thick vertex $\boldsymbol{u}$ of any thick graph, its neighbourhood $\mathrm{N}(\boldsymbol{u})$ must be a cluster graph, comprising subcliques of its thick neighbours. Thus its closed neighbourhood $\mathrm{N}[\boldsymbol{u}]$ is a unipolar graph. We use this observation to try to expand minimal clique separators to maximal clique separators of type (b), which contain a thick vertex.

We know already that we can recognise the superclass of quasi thick forests in $\mathcal{O}(mn)$ time using [51], so the first step is to verify that G is a quasi thick forest. If this fails, G is not a thick forest.

To decide if G is a thick forest, it suffices to decide whether each component of G is a thick tree, so we consider only the case where G is connected. We start with any maximal clique separator A of type (a) or (b), which we will propose as the root of the thick tree. Then the recognition problem will proceed by determining which of these two types applies to A. If neither holds then G is not a thick tree. We consider these two cases in 3.3.1 and 3.3.2.

We also require the following. If C is a clique such that $C \subseteq \boldsymbol{u}$ for some thick vertex $\boldsymbol{u}$, we will identify a maximal clique separator A such that $\boldsymbol{u} \subseteq A$, and A is of type (a) or (b). We show how this can be done efficiently in section 3.3.5. With this addition, we have the property that if the algorithm fails, it fails at the proposed root A of some subtree.

We also use expansion to initialise the algorithm, as follows. Choose any maximal clique separator A' in the clique cutset decomposition, choose $v \in A'$, and expand $C = \{v\}$ to give A. Clearly, $v \in \boldsymbol{u}$ for some thick vertex, so A will be of type (a) or (b), unless $\boldsymbol{u}$ is a leaf.

If $\boldsymbol{u}$ is in a leaf, then $\boldsymbol{uw}$ is a leaf edge for some $\boldsymbol{w}$. Then $A' \subset \boldsymbol{uw}$ is of the type considered in section 3.3.3 below, and we proceed as described there.

If the recognition algorithm succeeds, we will say that G is *accepted*, and we terminate with an explicit $H \in \mathsf{thin}(G) \cap \mathcal{F}$. Otherwise, $\mathsf{thin}(G) \cap \mathcal{F} = \emptyset$, and we will say that G is *rejected*.

Though our algorithm is general, we illustrate it with examples. The first will be the graph of Fig. 4, using the maximal clique separator $A = \{\mathsf{y},\mathsf{z},\mathsf{t}\}$. We reproduce it here for convenience.

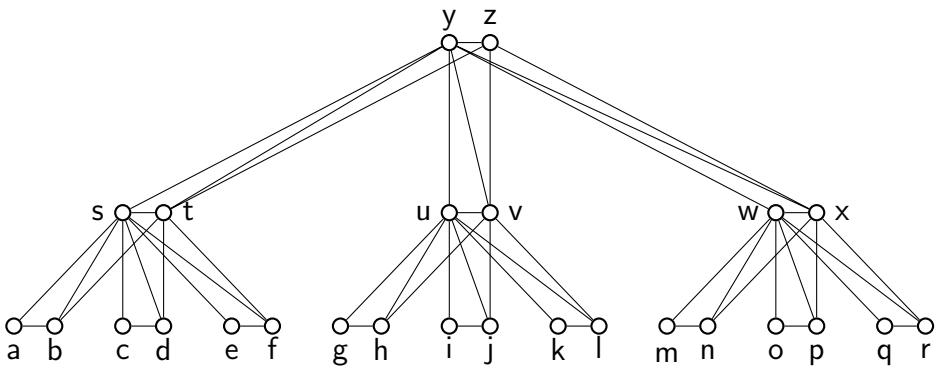


Fig. 4

3.3.1 A is of type (a): a thick vertex

If A is a thick vertex $\boldsymbol{u}$, either $V = \boldsymbol{u}$ or $V \setminus \boldsymbol{u}$ decomposes into disjoint sets $V_1, V_2, \ldots, V_r$ for some $r \geq 1$. If G is a thick tree, then each $G_i = G[V_i]$ must be a thick subtree for $i \in [r]$. Therefore the V_i can be constructed in $\mathcal{O}(m)$ time using, say, breadth-first search. Thus $\mathrm{N}(A)$ must be contained in a collection of disjoint cliques $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_r}$, where $\boldsymbol{uw}_i \in \mathcal{E}$ and $C_i = \mathrm{N}(A) \cap V_i \subset \boldsymbol{w}_i$ is a clique for $i \in [r]$. Then C_i is a minimal clique separator, separating $V_i \setminus C_i$ from $G \setminus V_i$. We expand C_i to a maximal clique separator A_i so that $C_i \subseteq \boldsymbol{w}_i \subseteq A_i$, using the method in section 3.3.5 below.. Then we determine if each of the subtrees G_i is a thick tree, starting with a maximal clique separator A_i containing the root $\boldsymbol{w}_i$. If all G_i are accepted as fat trees, we accept G.

There is a case where we can easily reject G_i. That is, if $\mathrm{N}(A) \cap V_i$ is not a clique. Otherwise, we must apply the algorithm to G_i.

In Fig. 4, if we assume A is a thick vertex, $G \setminus A$ has three components G_i ($i \in [3]$). In G_2 we have $C_2 = \{\mathsf{u},\mathsf{v}\}$ and in G_3, we have $C_3 = \{\mathsf{w},\mathsf{x}\}$. Both are cliques. However, in G_1, $\mathrm{N}(A) \cap V_1 = \{\mathsf{s},\mathsf{b},\mathsf{d},\mathsf{f}\}$ which is not a clique. Since not all the G_i are accepted, A is not type (a). But only G_1 is rejected, so we cannot conclude that G is not a thick tree. Instead, we must now suppose A is of type (b) and try to repair V_1.

3.3.2 A is of type (b): contains a thick vertex and is contained in a thick edge

In this case let A be contained in the unique thick edge $\boldsymbol{uw}_1$. Let $R = V_1$ and $L = \bigcup_{i=1}^{r} V_i$. Then L comprises $(r-1)$ thick trees with roots $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_r$, connected to G by thick edges $\boldsymbol{uw}_2, \ldots, \boldsymbol{uw}_r$. Note that L and R can be determined in $\mathcal{O}(m)$ time using graph search, and we assume this has been done.

Lemma 18. *Suppose, in a thick tree G, that the maximal clique separator A is such that $\boldsymbol{u} \subseteq A \subseteq \boldsymbol{uw}$ for some thick edge $\boldsymbol{uw}$. Then, in $O(m)$ time, we can determine a partition $\boldsymbol{u}, W$ of A and a clique W' such that $W \subseteq W' \subseteq \boldsymbol{w}$ and $G[\boldsymbol{u} \cup W']$ is a cobipartite graph and $\boldsymbol{u} : W'$ is a cut.*

Proof. We use the following algorithm, where each step has an explanatory comment. The comment is preceded by $\circ$.

`EDGE`

(0) Let $i \leftarrow 0$, $U_0 \leftarrow \{v \in A : \mathrm{N}(v) \cap L \neq \emptyset\}$, $U \leftarrow U_0$, $W' \leftarrow \emptyset$.
$\circ$ From above we have $U_0 \subseteq \boldsymbol{u}$, and $U_0 \neq \emptyset$ since A is a separator.

(1) $i \leftarrow i+1$, $W'_i \leftarrow (\mathrm{N}(U_{i-1}) \cap R) \setminus W'$, $W' \leftarrow W' \cup W'_i$.
$\circ$ Any edge $uw \in E$ with $u \in \boldsymbol{u}$, $w \in R$ must have $w \in \boldsymbol{w} \setminus A$. Note w cannot be in $\boldsymbol{u}$, since $\boldsymbol{u} \subseteq A$ and $A \cap R = \emptyset$.

(2) If W' is not a clique, stop: G is not a thick tree.
$\circ$ $W' \subseteq \boldsymbol{w}$ must be a clique if $\boldsymbol{uw}$ is a thick edge.

(3) $U_i \leftarrow (A \cap \bar{\mathrm{N}}(W'_i)) \setminus U$, $U \leftarrow U \cup U_i$.
$\circ$ Any non-edge $uw \notin E$ with $w \in \boldsymbol{w}$ and $u \in A$ must have $u \in \boldsymbol{u}$, since $u, w \in \boldsymbol{u} \cup \boldsymbol{w}$ and $\boldsymbol{w}$ is a clique. Note that $U_i \neq \emptyset$ if $W'_i \neq \emptyset$ since A is a maximal clique.

(4) If U is not a clique, stop: G is not a thick tree.
$\circ$ $U \subseteq \boldsymbol{u}$, so must be a clique.

(5) If $U_i \neq \emptyset$, return to step (1).
$\circ$ Otherwise W' and U are now maximal.

(6) $A' \leftarrow A \setminus U$.
$\circ$ Any $v \in A'$ must be complete to both U and W' from step (3) and $U \subseteq \boldsymbol{u} \subseteq A$.

(7) $W = \{v \in A' : \mathrm{N}(v) \cap R \not\subseteq W'\}$, $W' \leftarrow W' \cup W$.
$\circ$ Any vertex $w \in W$ is now complete to U and W', but is not mobile if it has a neighbour in $R \setminus W'$.

(8) $\boldsymbol{u} \leftarrow U \cup (A' \setminus W)$.
$\circ$ We have obtained $\boldsymbol{u}$ up to mobile vertices, since $\boldsymbol{u} \subseteq A$.

(9) Stop, the partition is $\boldsymbol{u}, W'$, with $W \subseteq W' \subseteq \boldsymbol{w}$.
$\circ$ Also $\boldsymbol{u} : W'$ is a cut in G since $\boldsymbol{u} \cup W'$ induces a cobipartite graph.

(10) The output $\boldsymbol{u} : W'$ of EDGE is a cut separating $\boldsymbol{u}$ from $G \setminus \boldsymbol{u}$, and we may expand W' to continue the recognition algorithm on $G \setminus \boldsymbol{u}$.

The time bound follows from the fact that the algorithm can be implemented so that it only examines any edge of G at most twice, once from each end vertex, given L and R. But determining these also requires $\mathcal{O}(m)$ time. □

Remark 10. Note that we only require A to be a separator in step (0), to ensure that $L \neq \emptyset$ and hence $U_0 \neq \emptyset$. If A is a maximal clique in $\boldsymbol{uw}$, and we have $\boldsymbol{s} \subseteq \boldsymbol{u} \subseteq A$ for some clique $\boldsymbol{s}$, then we can start the algorithm with $U_0 \leftarrow \boldsymbol{s}$.

In the recognition algorithm, the application of this is obvious. We determine the edge $\boldsymbol{u}W'$, so we have $\boldsymbol{u}$ and take $C_1 = W' \subseteq \boldsymbol{w}_1$. Then we determine whether $G_1 = G[V_1']$ is a thick tree, where $V_1' = (V_1 \cup A) \setminus \boldsymbol{u}$. Then we rerun the algorithm on G_1, with a proposed root A_1, obtained by expanding C_1 as described in section 3.3.5 below. However, we do not need to repeat the algorithm on any subtree that has been accepted in the first pass. Since only one subtree of any thick vertex can be rejected, the recomputations are actually restricted to a path in G. We try to repair this path bottom-up, starting at the point where a rejection first occurs. Then, if G_1 is accepted in this second pass, we accept G. Otherwise G is rejected. Note that we must take account of the accepted subtrees in the repairing the path, since this involves moving vertices between cliques. Hence the required reprocessing, using the algorithm of Lemma 18, does not adversely affect the time complexity of the algorithm.

In Fig. 4, we have $A = \{\mathsf{y,z,t}\}$. Since $\mathsf{y,z}$ have neighbours in R, so $U_0, U \leftarrow \mathsf{y,z}$, $W \leftarrow \emptyset$. Then $W_1', W' \leftarrow \mathsf{t}$, giving $U_1 \leftarrow \emptyset$. So $A' \leftarrow \mathsf{t}$, and $W \leftarrow \mathsf{t}, W' \leftarrow \mathsf{s,t}$. Thus $\boldsymbol{u} = \mathsf{y,z}$.

Thus $C_1 = W' = \{\mathsf{s,t}\} \subseteq \boldsymbol{w}_1$. We might expand $C_1 = \{\mathsf{s,t}\}$ to $A_1 = \{\mathsf{s,t,b}\}$, giving thick leaves $\{\mathsf{a}\}, \{\mathsf{c,d}\}, \{\mathsf{e,f}\}$. So we would verify that $G[V_1]$ is a thick tree. In G_2 we might expand $C_2 = \{\mathsf{u,v}\}$ to $A_2 = \{\mathsf{u,v,h}\}$, giving thick leaves $\{\mathsf{g}\}, \{\mathsf{i,j}\}, \{\mathsf{k,l}\}$. In G_3, we might expand $C_3 = \{\mathsf{w,x}\}$ to $A_3 = \{\mathsf{w,x,n}\}$, giving thick leaves $\{\mathsf{m}\}, \{\mathsf{o,p}\}, \{\mathsf{q,r}\}$. So we would accept all $G[V_i]$ ($i \in [3]$) and hence accept G.

As another example, consider the graph G of Fig. 5, which we reproduce below for convenience. The maximal clique separators are $\{\mathsf{b,h,g}\}$ and $\{\mathsf{e,h,g}\}$. Suppose we choose $A = \{\mathsf{b,g,h}\}$. This is not of type (a) since its neighbourhood is not a set of disjoint cliques. It is not of type (c) since it is not properly contained in two disjoint cliques. So it must be of type (b).

Then $V_2 = \{\mathsf{a}\}$, $V_3 = \{\mathsf{c}\}$, but $V_1 = \{\mathsf{d,e,f}\}$, a P_3 and not a clique. So V_1 must be repaired. But A has no mobile vertices, since g has neighbours in V_1, V_2, h has neighbours in V_1, V_3 and b has neighbours in V_2, V_3. Thus V_1 cannot be repaired and hence G is not a thick tree.

It is a minimal forbidden graph for $\mathcal{F}$ if deleting any vertex leaves a thick tree. By symmetry, there are only three cases to consider: a, b or g. Removing a or b gives a thick tree with $\{\mathsf{e,h,g}\}$ as root, removing g gives a thick tree with $\{\mathsf{h}\}$ as root. Thus G is a minimal forbidden subgraph for the class of thick forests, as claimed in section 3.2.

Finally, to illustrate the case where V_1 cannot be identified at the root, consider the graph of Fig. 12. Suppose $A = \{\mathsf{a,c,d}\}$, then in the left subtree, we get $C_1 = A_1 = \{\mathsf{b,f}\}$, and then the leaf $\{\mathsf{i,j}\}$. So this subtree is accepted. In the right subtree, we get $C_2 = A_2 = \{\mathsf{e,g}\}$, but this gives a P_3 $\{\mathsf{h,l,k}\}$ for the leaf, which is not a clique, so is rejected. So we know that V_1 is the right subtree and try repair it bottom-up. First we move g to give $\{\mathsf{g,h}\}$ and

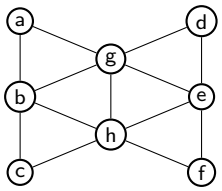


Fig. 5

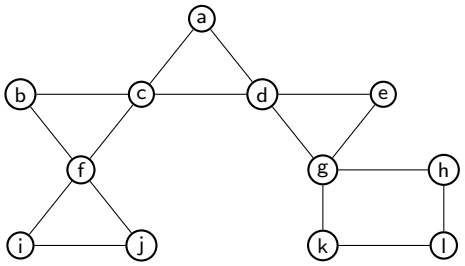


Fig. 12

a leaf $\{\mathsf{h},\mathsf{l}\}$. But now $\mathrm{N}(A) \cap V_1 = \{\mathsf{g},\mathsf{e}\}$ is no longer a thick vertex, so we must repair its remainder $\{\mathsf{e}\}$. We can move d into V_1, since it has no neighbours in the left subtree V_2, and we are done. This gives a thick vertex $\{\mathsf{d},\mathsf{e}\}$ and the root becomes $\{\mathsf{a},\mathsf{c}\}$. So G is a thick tree, with thick vertices, $\{\mathsf{a},\mathsf{c}\}$, $\{\mathsf{b},\mathsf{f}\}$, $\{\mathsf{i},\mathsf{j}\}$, $\{\mathsf{d},\mathsf{e}\}$, $\{\mathsf{g},\mathsf{h}\}$ and $\{\mathsf{k},\mathsf{l}\}$.

3.3.3 A is of type (c) and is contained in a thick leaf edge

We will need to consider maximal clique separators of type (c) only in the case where the maximal clique separator $A \subset \boldsymbol{uw}$ is such that $\boldsymbol{u}$ is a leaf of $\mathsf{thin}(G)$. We recognise this as described in step (0) of `LEAF` below, by reducing to the case considered in section 3.3.2. We apply the following algorithm, using the same format as Lemma 18.

`LEAF`

(0) Remove A from G, giving a clique $\boldsymbol{s} \leftarrow \boldsymbol{u} \setminus A$, and a connected graph $G' = G \setminus \boldsymbol{s}$. If not, stop $\boldsymbol{u}$ is not a leaf of a thick tree. Otherwise, $R \leftarrow V(G' \setminus A)$.
∘ This requires $\mathcal{O}(m)$ time, using a graph search. Note that $\boldsymbol{s}$ is a known subset of $\boldsymbol{u}$, and both $\boldsymbol{s}, R \neq \emptyset$, since A is a separator.

(1) Let $B \leftarrow \{v \in A : v \text{ is complete to } \boldsymbol{s}\}$, $A' \leftarrow \boldsymbol{s} \cup B$.
∘ B is a clique, since $B \subseteq A$. Then A' is a clique since B is complete to the clique $\boldsymbol{s}$. Also A' is a maximal clique. Otherwise, suppose $v \notin A'$ is complete to A'. If $v \in \boldsymbol{s}$, then $v \in A'$ contradicting $v \notin A'$. If $v \in A$, then $v \in B$, contradicting $v \notin A'$. If $v \notin A$, then $vs \notin E$ for some $s \in \boldsymbol{s}$, since A separates $\boldsymbol{s}$, contradicting v being complete to A'. Finally, we have $\boldsymbol{u} \subseteq A'$. This follows from $\boldsymbol{u} \setminus A = \boldsymbol{s}$ and $\boldsymbol{u} \cap A \subseteq B$, since every vertex in $\boldsymbol{u} \cap A$ is complete to $\boldsymbol{u}$ and hence to $\boldsymbol{s}$. Thus $\boldsymbol{u} = (\boldsymbol{u} \setminus A) \cup (\boldsymbol{u} \cap A) \subseteq \boldsymbol{s} \cup B = A'$.

(2) Run the algorithm `EDGE`, with $A \leftarrow A'$, $U_0 \leftarrow \boldsymbol{s}$.
∘ This is justified by Remark 10.

(3) The output $\boldsymbol{u} : W'$ of `EDGE` is a cut separating the leaf $\boldsymbol{u}$ from $G \setminus \boldsymbol{u}$, and we may

expand $W' \subseteq \boldsymbol{w}$ to continue the recognition algorithm.

Step (0) clearly takes $\mathcal{O}(m)$ time, and (2) takes $\mathcal{O}(m)$ time from Lemma 18. Thus we have the following.

Lemma 19. *Suppose, in a thick tree G, that the maximal clique separator $A \subset \boldsymbol{uw}$ is such that $A \setminus \boldsymbol{u}, A \setminus \boldsymbol{w} \neq \emptyset$ for some thick edge $\boldsymbol{uw}$ with $\boldsymbol{u}$ a leaf of $\mathsf{thin}(G)$. Then, in $O(m)$ time, we can determine $\boldsymbol{u}$ and a clique $W' \subseteq \boldsymbol{w}$ such that $\boldsymbol{u} : W'$ is a cut separating $\boldsymbol{u}$ from $G \setminus \boldsymbol{u}$.*

Observe that identification and deletion of a thick leaf in this way may give an alternative approach to thick tree recognition. However, we will not pursue this further here.

Consider the example in Fig. 13, where $A \leftarrow \{\mathsf{f}, \mathsf{g}, \mathsf{i}, \mathsf{j}\}$, $R \leftarrow \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$ and $\boldsymbol{s} \leftarrow \{\mathsf{h}\}$. Running `LEAF` gives $B \leftarrow \{\mathsf{f}, \mathsf{i}, \mathsf{j}\}$, so $A' \leftarrow \{\mathsf{f}, \mathsf{h}, \mathsf{i}, \mathsf{j}\}$. Then `EDGE` gives $\boldsymbol{u} \leftarrow \{\mathsf{h}, \mathsf{i}, \mathsf{j}\}$, $W' \leftarrow \{\mathsf{f}, \mathsf{g}\}$, since f has neighbours $\{\mathsf{b}, \mathsf{c}, \mathsf{e}\} \in R$, and the cut $\boldsymbol{u} : W'$ shown.

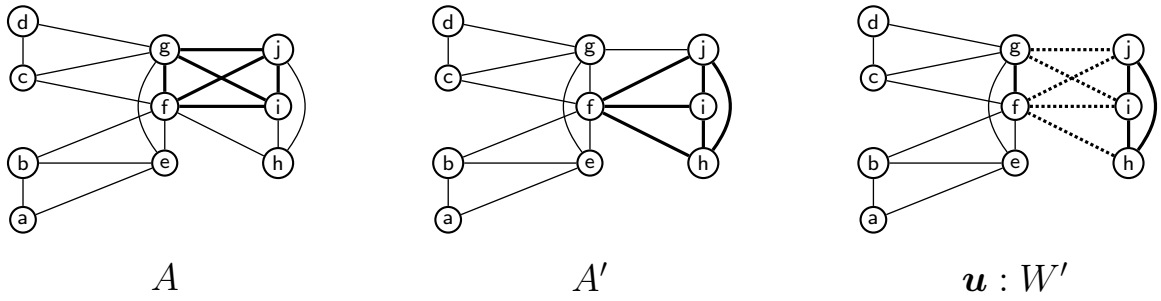


Fig. 13: Thick leaf edge case

3.3.4 Recognising a unipolar graph

We can now show how our methods may be used to recognise a unipolar graph, a thick star. Let G be a unipolar graph with a thick vertex *hub* $\boldsymbol{h}$ and thick leaves $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_s$, the *satellites*. This can be done in $\mathcal{O}(n^3)$ time using the algorithm of [24] or that of [52], but we will now show how to do this using our methods.

`UNIPOLAR`

(1) Find a clique cutset decomposition of G_C using the algorithm of [51], giving r maximal clique separators.
∘ Note that $r \leq s + 1 \leq n$.

(2) If $r = 0$, test whether G is a clique or a cobipartite graph and stop. If so, G is unipolar. If not, G is not unipolar. If G is cobipartite on $\boldsymbol{u} \cup \boldsymbol{w}$, we may take $\boldsymbol{h} \leftarrow \boldsymbol{u}$ and $\boldsymbol{b}_1 \leftarrow \boldsymbol{w}$ or vice versa.
∘ Cliques and cobipartite graphs are the only unipolar graphs with no maximal clique separator. Cliques have none, but cobipartite graphs can have them. See Lemma 13.

(3) Otherwise, number the maximal clique separators $A_1, \ldots, A_r$ arbitrarily. For each of the r in order test whether the the removal of A_i decomposes G into disjoint cliques. If so, then $\boldsymbol{h} \leftarrow A_i$ is a hub for G and $G \setminus \boldsymbol{h}$ is a cluster graph giving the satellites.

∘ Any maximal clique separator in G of type (a) or (b) must contain the hub $\boldsymbol{h}$ of G, or it would not be a separator. Thus any such maximal clique separator can be taken as the hub if its removal decomposes G into satellites. Thus we can test this in $\mathcal{O}(mr) = \mathcal{O}(mn)$ time, by testing that all components are cliques.

(4) Otherwise $s \leftarrow 1$, $G_1 \leftarrow G$.

(5) For the maximal clique separator A_s, run `LEAF` to give a cut $W'_s : \boldsymbol{b}_s$.
∘ All the maximal clique separators must be of type (c) and contained in leaf edges if G is unipolar. Note that R must always be determined with respect to G, not G_s, in step (7) of `LEAF`, to avoid hub vertices being incorrectly moved into satellites.

(6) Delete the edges of $W'_s : \boldsymbol{b}_s$ from G_s giving $G_{s+1} \leftarrow G_s \setminus \boldsymbol{b}_s$.
∘ G_{s+1} is a unipolar graph with satellites $\boldsymbol{b}_{s+1}, \boldsymbol{b}_{s+2}, \ldots$ if these are nonempty.

(7) If G_{s+1} is not a clique, set $s \leftarrow s + 1$ and return to step (5).

(8) $\boldsymbol{h} \leftarrow G_{s+1}$ is the hub, and $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s$ are the satellites.
∘ Note that now $s = r - 1$.

If the algorithm fails, it must fail in step (5). Each repetition of the loop requires $O(m)$ time, so the total time required is $\mathcal{O}(ms) = \mathcal{O}(mn)$, which slightly improves the $\mathcal{O}(n^3)$ in [24] or [52]. Thus we have the following.

Lemma 20. *If G is a unipolar graph, in $\mathcal{O}(mn)$ time we can determine a hub $\boldsymbol{h}$ and satellites $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_s$.*

Remark 11. In fact, the time bound in [24] is not given as $\mathcal{O}(n^3)$, but depends on the number of edges in a minimal triangulation of G. It seems that a more careful analysis could yield the same $\mathcal{O}(mn)$ given here. Note also that the above algorithm depends indirectly on triangulation through Tarjan's algorithm [51].

As an example, note that the graph of Fig. 13 is unipolar. With $s = 1$ we might remove $\{\mathsf{h}, \mathsf{i}, \mathsf{j}\}$, as shown in Fig. 13, $\{\mathsf{c}, \mathsf{d}, \mathsf{g}\}$. If the next maximal clique separator is $\{\mathsf{c}, \mathsf{d}, \mathsf{g}\}$, then `LEAF` would determine the cut $\{\mathsf{c}, \mathsf{d}\}$:$\{\mathsf{f}, \mathsf{g}\}$, giving $G_3 = G[\{\mathsf{a}, \mathsf{b}, \mathsf{e}, \mathsf{f}, \mathsf{g}\}]$. The final maximal clique separator is then $\{\mathsf{a}, \mathsf{b}, \mathsf{g}\}$. Then `LEAF` would determine the cut $\{\mathsf{a}, \mathsf{b}\}$:$\{\mathsf{e}, \mathsf{f}\}$, giving $G_4 = G[\{\mathsf{e}, \mathsf{f}, \mathsf{g}\}]$ which is a clique. Thus the decomposition of G has hub $\{\mathsf{e}, \mathsf{f}, \mathsf{g}\}$ and satellites $\{\mathsf{a}, \mathsf{b}\}$, $\{\mathsf{c}, \mathsf{d}\}$, $\{\mathsf{h}, \mathsf{i}, \mathsf{j}\}$.

3.3.5 Expanding a clique C to a maximal clique separator A

We have a clique $C \subseteq \boldsymbol{u}$ for some thick vertex $\boldsymbol{u}$, and we wish to find a maximal clique A so that $C \subseteq \boldsymbol{u} \subseteq A$. Then A is an maximal clique separator of type (b), since it is a maximal clique separator and contains $\boldsymbol{u}$.

To do this, we construct the graph $G_C = (V_C, E_C)$ induced by V_C, the set of vertices in G which are either in C or are complete to C. Then any clique in G containing C is also a clique in G_C. Clearly $C \subseteq \boldsymbol{u} \subseteq V_C$. Moreover, $\boldsymbol{u}$ is a clique separator in G and hence also in G_C. Thus any maximal clique in G_C which includes $\boldsymbol{u}$ is a maximal clique separator in G. However, G_C may also have vertices from the neighbours $\boldsymbol{w}_i$ ($i \in [r]$) of $\boldsymbol{u}$ in $H = \mathsf{thin}(G)$,

where $r \leq \deg_H(\boldsymbol{u})$. These also induce cliques in G_C, since $\boldsymbol{w}_i$ is a clique and all vertices in V_C are complete to C.

Thus G_C is a unipolar graph with hub $\boldsymbol{h}$ and satellites $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_s$, where $\boldsymbol{b}_i$ is contained in a thick neighbour $\boldsymbol{w}_i$ of $\boldsymbol{h}$, for $i \in [s]$ and $s \leq r$. Any maximal clique separator not properly contained in a thick edge of G_C must contain a thick vertex of G_C, and this must contain the thick vertex $\boldsymbol{u}$ of G. Thus determining $\boldsymbol{h}$ yields a maximal clique separator of G of type (a) or (b).

Though G_C is close to being chordal, it is not necessarily so. Edges between the satellites $\boldsymbol{b}_i$ and $\boldsymbol{u} \setminus C$ can induce a C_4, so we must treat G_C as a general unipolar graph. Consider, for the case where C is a single vertex v. Then G_C is simply the graph induced by $\mathrm{N}[v]$, which need not be chordal. An example graph G is shown in Fig. 14. If $C = \{\mathsf{c}\}$ then $G_C = G$ and has a 4-hole $G[\mathsf{a},\mathsf{d},\mathsf{e},\mathsf{b}]$. Note that G is cobipartite, so is unipolar. There are four possibilities for the hub, the four triangles around c.

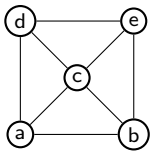


Fig. 14: Nonchordal G_C

We apply the algorithm `UNIPOLAR` to G_C, and set $A \leftarrow \boldsymbol{h}$, the hub. If G_C has n_C vertices and m_C edges, the overall time bound for decomposing G_C is $\mathcal{O}(m_C n_C)$ from Lemma 20. This is $\mathcal{O}(mn)$, so does not affect the time analysis of section 3.3.6 below.

Note that there can be more than one possibility for $\boldsymbol{h}$, but at most $s \leq n_C$. These are all maximal clique separators in G_C. They can be determined from the unipolar decomposition, but we must check that they separate G since we want a maximal clique separator of type (b). We can check for each in $\mathcal{O}(m)$ time by graph search, so the overall time bound is again $\mathcal{O}(mn)$. We can choose any which passes this check as the required maximal clique separator but G is not a thick tree if none does. This is clear, we have found a thick vertex $\boldsymbol{u}$ which does not separate G. Note that $\boldsymbol{u}$ cannot be a leaf since then we would use `LEAF` rather than `UNIPOLAR`.

This completes the description of the algorithm.

Consider the fragment of a graph G in Fig. 15, where $C = \{\mathsf{a},\mathsf{b}\}$ and $\boldsymbol{u} = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$. The vertices complete to C are $\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e},\mathsf{h}\}$. The induced graph G_C is also shown in Fig. 15. Here G_C is actually a chordal graph, which has two maximal clique separators, $\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d}\}$ and $\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{h}\}$. Both $\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d}\}$ and $\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{h}\}$ include $\boldsymbol{u}$ and are separators in G, so either could be used as a maximal clique separator A of type (b) in G.

As a illustration of the necessity of checking the hub for separation, consider the graph of Fig. 16 with $C = \mathsf{b}$. Then G_C has three possibilities for its hub $\{\mathsf{a},\mathsf{b}\}$, $\{\mathsf{b},\mathsf{c}\}$, $\{\mathsf{b},\mathsf{e}\}$. The first two do not separate G but the third does. So we would proceed with $A \leftarrow \{\mathsf{b},\mathsf{e}\}$. This leads to a thick tree decomposition for G.

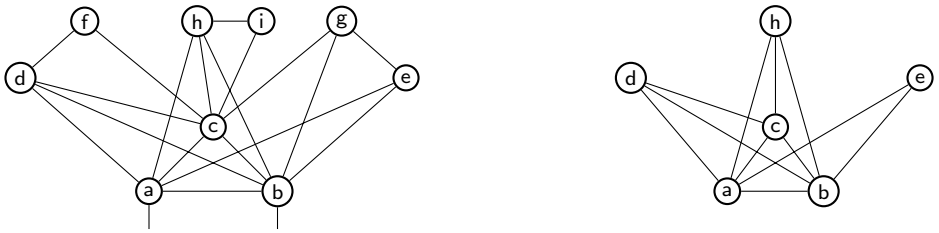


Fig. 15: Fragment of a graph G with $C = \{\mathsf{a},\mathsf{b}\}$ and G_C

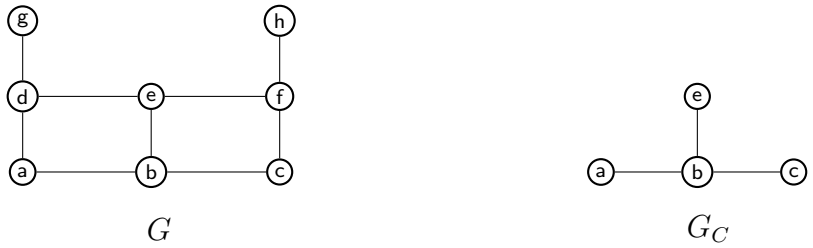


Fig. 16: Graph G with $C = \{\mathsf{b}\}$ and G_C

3.3.6 Running time analysis

We now determine the time and space complexity of the algorithm. The space required is simply $\mathcal{O}(m)$, as in the algorithm of [51]. We start with the algorithm of [51], which requires $\mathcal{O}(mn)$ time. We process a maximal clique separator A which generates $r \geq 1$ subtrees. The time to carry this out is $\mathcal{O}(m)$. We might have to process A both for a thick vertex and a thick edge, but the total will still be $\mathcal{O}(m)$. If A is in a thick edge, then $r = 2$. Then we check r subtrees with n_i vertices and m_i edges for $i \in [r]$. The total time for generating the A_i from C_i has been bounded by $O(m_i n_i)$ above.

We claim that the recognition algorithm has time complexity $T(m,n) = \mathcal{O}(mn)$. This is easily seen to be true for atoms, which are cliques or cobipartite graphs. Then, since $m_i + n_i < m + n$ for $i \in [r]$, the time complexity is bounded by

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

Thus the overall time complexity of recognising thick forests is $\mathcal{O}(mn)$, which is larger than the bound for recognising quasi thick forests by only a constant factor.

4 Counting independent sets

We use the clique cutset decomposition approach of [22, Sec. 2.2], so the algorithm applies to quasi thick forests. The paper [22] concerns counting independent sets in claw- and fork-free perfect graphs. 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 T is a path, and can contain a fork unless the thin tree T 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 [22]. Note that a claw or a fork, in fact any tree, does not fall within the scope of Lemma 4 since it has a clique cutset of size one.

As in [22], we consider counting weighted independent sets. That is, evaluating the 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 $W(G) = \sum_{S\in\mathcal{I}(G)} w(S) = \sum_{k=0}^{n} W_k(G)$. In particular, we have $W_0(G) = 1$, $W_1(G) = \sum_{v\in V} w(v)$ and $W_2(G) = \sum_{uv\notin E} w(u)w(v)$.

The difference from [22] is only in the atoms of the clique cutset decomposition. Here they are cliques and cobipartite graphs. For these we can evaluate the W_k exactly in linear time. Thus $W_0 = 1$, $W_1 = \sum_{v\in K} w(v)$, $W_k = 0$ $(k > 1)$ for a clique K. 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 a quasi thick forest G, and there is no need to introduce the error control used in [22, Sec. 2.2.1]. For further details, see [22]. Thus we have a deterministic algorithm for counting weighted independent sets in thick forests.

A consequence of the exact evaluation of $W(G)$ is that we can evaluate $W_k(G)$ exactly for all $0 \leq k \leq n$ by interpolation. The weighted independence polynomial is $P(\lambda) = \sum_{k=0}^{n} W_k\lambda^k$, and the W_k are its coefficients. 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 α coefficients W_k $(0 < k \leq \alpha)$.

5 Counting colourings

Again we use clique cutset decomposition in a similar way to [22], but there we would not count colourings since we cannot count them in the atoms that are line graphs of bipartite graphs. That is, we cannot counting edge-colourings of bipartite graphs, even approximately.

We show below, in section 5.1, that exact counting is #P-complete for thick trees, since it is #P-complete for thick edges. So we can only count colourings approximately. Again, we refer the reader to [22, Sec. 2.2.1] for details of error control in clique cutset decomposition.

The algorithm we give here is based on the following *clique cutset colouring lemma* (CCCL).

Lemma 21 (CCCL)**.** *Suppose $G = (V, E)$ and $K \subseteq V$ is a clique cutset of size $k = |K|$ in G. Let $G_i = G[V_i]$ ($i \in [2]$). Then $\#\mathsf{col}_q(G) = \#\mathsf{col}_q(G_1)\#\mathsf{col}_q(G_2)/(q)_k$.*

Proof. K has $(q)_k$ colourings, each corresponding to a permutation of a selection of $\binom{q}{k}$ 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 $\#\mathsf{col}_q(G_1)\#\mathsf{col}_q(G_2)$, each colouring of G is counted $(q)_k$ times, so we correct for this overcounting. We do not have to assume G is connected if we allow $K = \emptyset$ as a cutset of size 0. □

Note that the proof of the CCCL relies completely on the symmetries of both q-colouring and cliques, thus it does not seem to have any easy generalisations.

Thus, if we can compute (an approximation to) $\#\mathsf{col}_q(A)$ for each atom of the clique cutset decomposition tree, we can use the CCCL recursively in an obvious fashion to obtain (an approximation to) $\#\mathsf{col}_q(G)$. See [22] for further details.

Thus we require (an approximation to) the number of colourings of an atom of the decomposition tree for a thick forest. For quasi thick forests, these are cliques and cobipartite graphs. There are exactly $(q)_k$ q-colourings for a clique of size k, so we need only consider the atoms which are cobipartite graphs.

5.1 Cobipartite graphs

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 in $\{v_1, v_2\} \in E_{12}$.Thus vertices in V_1, V_2 can only receive the same colour if they form a *matching* in the bipartite graph $\bar{B}$. 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, we have

$$\#\mathsf{col}_q(G) = \sum_{k=n-q}^{n} \kappa_k(\bar{B})(q)_{n-k}. \tag{5.1}$$

As a function of q, this is the *chromatic polynomial* of G. In particular, if $n_1 = n_2 = q$, this becomes $(q)_q\kappa_q = q!\kappa_q$. Since κ_q is then the number of *perfect* matchings, we have

Lemma 22. *Counting the number of q-colourings of a cobipartite graph is #P-complete.*

Proof. The above is a reduction from counting the number of perfect matchings in a bipartite graph. Then Valiant's proof [53] that this is #P-complete implies the result. □

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 [36]. (This algorithm was used for a similar purpose in [22].) Since the chromatic polynomial is a linear function of the numbers of k-matchings with positive coefficients, we have an FPRAS for $\#\mathsf{col}_q(G)$ for any q on any cobipartite graph. We can then use clique cutset decomposition to give an FPRAS for $\#\mathsf{col}_q(G)$ for any q on a quasi thick forest.

Note that if G is a chordal graph, there are no cobipartite atoms, so no need for approximation, and we have an exact algorithm for counting q-colourings of chordal graphs, which can be implemented to run in $\mathcal{O}(m+n)$ time. However, the chromatic polynomial for a chordal graph can be determined in $\mathcal{O}(m+n)$ time using a perfect elimination order. See [1, Rem. 2.5]. This can also be implemented to run in $\mathcal{O}(m+n)$ time for a chordal graph. Recognition can also be done in $\mathcal{O}(m+n)$ time, so counting q-colourings is in $\mathcal{O}(m+n)$ time for a chordal graph using either approach. However, since the number of colourings can be as large as q^n, an additional $\mathcal{O}(\log q)$ factor is required in the bit-complexity model.

6 Beyond thick forests

6.1 Recognition

We have seen that whether $G \in \mathsf{thick}(\mathcal{F})$ is decidable in P for the class of forests $\mathcal{F}$. An obvious question is whether this is true for any other class $\mathcal{C}$.

The case where $\mathcal{C} = \{H\}$ for any fixed graph was resolved completely by MacGillivray and Yu [43]. They proved the following, which we recast in our own notation. Recall that $\mathcal{T}$ is the class of triangle-free graphs.

Theorem 2 (MacGillivray and Yu)**.** *If $H \in \mathcal{T}$, the decision problem $G \in \mathsf{thick}(H)$ is in P. Otherwise it is NP-complete.* □

However, the recognition problem given in [43] is exponential in $\nu = |V(H)|$. It is an interesting question whether this can be improved to an algorithm in FPT, with parameter ν. This has been done for thick bipartite graphs in [39], where an FPT algorithm is given, even with the stronger parameterisation that ν is the size of the smaller part of the bipartition. The algorithm has time complexity $\mathcal{O}(\nu^{2\nu+1}mn)$, and can be improved to $\mathcal{O}(4^k k^2 b^2)$ if ν is the size of the larger part of the bipartition. This is another direction of generalisation of the result for unipolar graphs than to $\mathcal{F}$. It seems that the technique developed in [39] may give a positive answer to the above question regarding the algorithm of [43].

The criterion of Theorem 2 clearly extends to any finite class $\mathcal{C} = \{H_1, H_2, \ldots, H_k\}$, simply by testing each H_i in turn. This idea also extends to the case where $\mathcal{C}_n = \{G \in \mathcal{C} : |V| \leq n\}$ is of size only polynomial in n provided, of course, that we can list the graphs in $\mathcal{C}_n$ in time polynomial in n.

However, the criterion of Theorem 2 does not extend to infinite classes in general, though clearly $\mathsf{thick}(\mathcal{C})$ cannot be recognisable for any class $\mathcal{C}$ not contained in $\mathcal{T}$. However, we

have seen that recognition of $G \in \mathsf{thick}(\mathcal{F}) \subset \mathsf{thick}(\mathcal{T})$ (thick forests) is in P, but $G \in \mathsf{thick}(\mathcal{B}) \subset \mathsf{thick}(\mathcal{T})$ (thick bipartite graphs) is NP-complete. We leave as an open question what properties are required of $\mathcal{C}$ in general for $G \in \mathsf{thick}(\mathcal{C})$ to be recognisable in P.

In this respect, it should be observed that if $G \in \mathsf{thick}(H)$, G may have very different properties from H. For example, bipartite graphs are perfect, but a thick bipartite graph is not necessarily perfect. See Fig. 17. More obviously, a tree has no cycles, but a thick tree may have an exponential number, the most extreme example being K_n.

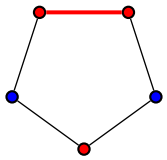


Fig. 17: C_5 as a thick C_4

6.2 Colourings

We will show that approximately counting colourings of a thick graph with any thin graph which is not a forest is NP-hard. First we show that

Lemma 23. *If $\triangle$ is a triangle, it is NP-complete to determine if $T \in \mathsf{thick}(\triangle)$ has a proper q-colouring.*

Proof. We consider q-colouring a $T = (V, E) \in \mathsf{thick}(\triangle)$ with three K_q thick vertices $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$. (See Fig. 18.) Thus each of the colours must occur exactly once in each thick vertex. Observe then that G 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. □

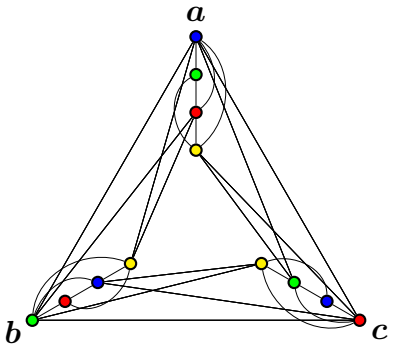


Fig. 18: A proper 4-colouring of a thick triangle with K_4 thick vertices

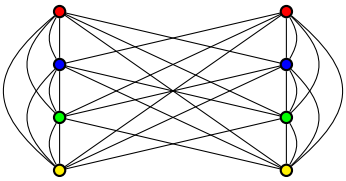


Fig. 19: A copy thick edge with $q = 4$

We can extend this using *copy* thick edges. 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 are those which duplicate the colourings on the right and left sides.

Then we have

Lemma 24. *For any $k \geq 3$, it is NP-complete to determine if $G \in \mathsf{thick}(C_k)$ has a proper q-colouring,.*

Proof. f $k = 3$, then this is Lemma 23. If $k > 3$, consider a thick triangle $T \in \mathsf{thick}(\triangle)$ with thick K_q vertices $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, as in the proof of Lemma 23. We replace one of the thick edges of T, $\boldsymbol{ab}$ say, by a $(k-2)$ thick path of K_q vertices with $(k-3)$ copy edges and a copy of $\boldsymbol{ab}$. This is a thick k-cycle, and clearly has a proper q-colouring if and only if T has a proper q-colouring. □

Lemma 25. *If $H \notin \mathcal{F}$, the class of forests, it is NP-complete to determine if $G \in \mathsf{thick}(H)$ has a proper q-colouring.*

Proof. We construct G as follows. All thick vertices of G will be K_q's, and there is an underlying $T \in \mathsf{thick}(\triangle)$ with thick K_q vertices $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$. If $H \notin \mathcal{F}$, it has a cycle. Let the shortest cycle in H have length k. Replace this cycle in G by the thick k-cycle C' from Lemma 24. We use the thick vertices $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ of T to label the thick vertices of G and we will associate these labels with the colourings in T. So $(k-2)$ vertices of C' are labelled $\boldsymbol{a}$, one is labelled $\boldsymbol{b}$ and one is labelled $\boldsymbol{c}$. Then we construct the remainder of G from H as follows. Choose a thick vertex $\boldsymbol{v}$ labeled $\boldsymbol{x} \in \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, and consider any neighbour $\boldsymbol{w}$. If $\boldsymbol{w}$ is unlabelled, label it $\boldsymbol{x}$ and make $\boldsymbol{vw}$ a copy edge. If $\boldsymbol{w}$ is also labeled $\boldsymbol{x}$, again make $\boldsymbol{vw}$ a copy edge. If $\boldsymbol{w}$ is labeled $\boldsymbol{y} \neq \boldsymbol{x}$, then make $\boldsymbol{vw}$ a copy of the thick edge $\boldsymbol{xy}$ of T. We continue until all thick edges have been assigned. Now it is clear that G has a proper q-colouring if and only if T has a proper q-colouring. □

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

Theorem 3. *If $\mathcal{C}$ is any class of graphs such that $\mathcal{C} \not\subseteq \mathcal{F}$, then counting proper q-colourings of $G \in \mathsf{thick}(\mathcal{C})$ is complete for* #P *under* AP*-reducibility.*

Proof. Follows directly from Lemma 25 and [20, Thm. 1]. □

Hence our results for approximately counting colourings of thick forests do not extend to any larger class of thick graphs unless P = NP.

6.3 Graphs of bounded treewidth

Treewidth is one of the most important parameters for graphs because of its uses in algorithmic meta-theorems and the theory of graph minors [18]. Below we recall the definition via tree decompositions, but equivalent definitions exist [33]. For further information and references to original work see [8, 10]. Here we give two applications of this to thick graphs.

Treewidth measures how treelike a graph is. Let $\mathcal{W}_k$ be the class of graphs with treewidth at most k. So $\mathcal{W}_1 = \mathcal{F}$, and we have shown that $\mathsf{thick}(\mathcal{F})$ is a very tractable class of graphs. Here we consider $\mathsf{thick}(\mathcal{W}_k)$ when $k > 1$. These classes are all intractable, since $\mathcal{W}_2$ includes the triangle, and so from Theorem 2 recognition in $\mathsf{thick}(\mathcal{W}_k)$ is intractable for all $k \geq 2$. However, we will show that we can exactly count independent sets in $\mathsf{thick}(\mathcal{W}_k)$ if we assume recognition.

Finally, we saw in section 5 that exact counting of colourings in thick trees is #P-complete, even for a single thick edge, but there is an FPRAS. Here we will show that, if we are guaranteed that G can be represented as a thick forest with thick vertices of bounded size, then we can count colourings deterministically. Thus, if we let $\mathcal{F}_b$ be the class of thick forests with all thick vertices of size at most b, we show deterministic counting for $\mathsf{thick}(\mathcal{F}_b)$. We do this by bounding the treewidth of graphs in $\mathsf{thick}(\mathcal{F}_b)$. Note that the treewidth of a thick tree of size n can be $(n-1)$, since it can be an n-clique, so this approach requires a bound on the size of thick vertices.

6.3.1 Definitions

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 all vertices $v \in V$ exist an $i \in I$ such that $v \in b(i)$,
2. for all edges $e \in E$ exist an $i \in I$ such that $e \subseteq b(i)$, and
3. for all vertices $v \in V$ 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 $i \in I$, 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 $\mathrm{tw}(G)$, 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 $\mathrm{tw}(K_n) = n - 1$. For a tree G with at least one edge, the subdivision mentioned above is optimal, so $\mathrm{tw}(G) = 1$.

Computing the treewidth of a graph is NP-hard, even when restricted to co-bipartite graphs [2]. On the other hand, the problem is in FPT if parameterised by the treewidth [7, 9].

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 $\mathrm{tw}(G)$ 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 An 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 [40]. Every graph $G = (V, E)$ has a nice tree decomposition (T, b) of minimum width such that $|I| = \mathcal{O}(|V|)$.

6.3.2 Independent sets

Here we give a deterministic algorithm computing the number $\#\mathsf{ind}(G)$ of independent sets of a thick graph G given together with a model (H, ψ) with $\mathrm{tw}(H) \leq k$ for some constant $k > 1$. Thus we are assuming that recognition of $G \in \mathsf{thick}(\mathcal{W}_k)$ is given.

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

Lemma 26. *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 $\boldsymbol{v} \in b(i)$ corresponds to a clique $\psi^{-1}(\boldsymbol{v})$ of G by property 1 of the model (H, ψ). □

Let (T, b) be a nice tree decomposition of H. Let r be root of $T = (I, A)$ and let k denote the width of (T, b). 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 $G(i) = G[\psi^{-1}\big(\bigcup_{j \in C(i)} b(j)\big)]$ that coincide with S on $\psi^{-1}(b(i))$. That is, $\#\mathsf{ind}(G(i)) = \sum_S a(i, S)$, 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 $\boldsymbol{w} \in b(i) \setminus b(j)$. For every independent set $S \subseteq \psi^{-1}(b(j))$ and every vertex $v \in \psi^{-1}(\boldsymbol{w})$ 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 $\boldsymbol{w}$ 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 $\boldsymbol{w} \in b(j) \setminus b(i)$. For every independent set $S \subseteq \psi^{-1}(b(j))$ and every $v \in \psi^{-1}(\boldsymbol{w})$ we have $a(i,S) = a(j,S)$ if $v \notin S$ and $a(i, S \setminus \{v\}) = a(j, S \setminus \{v\}) + a(j,S)$ 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 $\#\mathsf{ind}(G) = a(r, \emptyset)$ independent sets because $S \cap b(r) = \emptyset$ holds for all independent sets S of G.

For a graph G on n vertices 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 26 the independent set S chooses at most $k+1$ vertices from the set $\psi^{-1}(b(i))$. Consequently, the algorithm runs in time $\mathcal{O}(n^{k+2})$, which is $\mathcal{O}(n^3)$ for thick trees. It can easily be extended to deal with weighted independent sets, with a similar analysis.

Thus we have an algorithm in XP, but we leave as an open the question whether there is an FTP algorithm.

6.3.3 Colourings

We consider counting q-colourings in $\mathcal{F}_b$, but we express this as counting q-colourings in $\mathcal{W}_{4b-1}$ as follows.

Lemma 27. *If $G \in \mathsf{thick}(\mathcal{F}_b)$, then we can find a tree decomposition of G width w with $b-1 \leq w \leq 4b-1$ in time $\mathcal{O}(b^2 n \log n)$.*

Proof. First we use the algorithm of section 3.3 to find a thick tree representation of G in time $\mathcal{O}(mn)$. Now note that any thick tree representation of $G \in \mathsf{thick}(\mathcal{F}_b)$ is in $\mathsf{thick}(\mathcal{F}_{2b})$. This follows from the fact that any thick vertex can only be expanded by moving vertices from exactly one of its thick neighbours, and thus all thick vertices can have size at most $2b$. Now a tree decomposition can be obtained from a representation of a thick tree having thick vertices of size at most s by using the thick edges as bags, and the thick tree structure in the obvious way. So all bags have size at most $2s$ and hence the decomposition has width at most $2s-1$. Putting $s = 2b$ gives the upper bound. The lower bound follows since G contains a clique C of size at least b if $G \notin \mathsf{thick}(\mathcal{F}_{b-1})$, and hence $\mathrm{tw}(G) \geq \mathrm{tw}(C) \geq b-1$.

For the time bound, note that $m = \Theta(b^2 n)$, using Lemma 8. Then, if $b = o(n)$, Tarjan's algorithm can be used find in $O(m)$ time an initial maximal clique separator for the algorithm of section 3.3 which divides the graph roughly into two, so we have running time $t(n) = 2t(n/2) + O(b^2 n)$. This is a standard recurrence, with the solution claimed. □

Remark 12. Since $\mathcal{F}_b \subseteq \mathcal{W}_{2b-1}$, we could have used an existing algorithm for approximating the tree decomposition. See [7] for a survey of recent developments. However, these algorithms would not run in time polynomial in b. That is because these algorithms also rely on computing separators, but their greater generality means that this cannot be done as efficiently as for $\mathcal{F}_b$. However, these algorithms correctly identify (say) $G \in \mathcal{W}_{5s}$ for any $G \in \mathcal{W}_s$, and not just $G \in \mathcal{W}_{4b-1}$ for any $G \in \mathcal{F}_b$.

Now let (T, b) be a tree decomposition of width k for a graph $G = (V, E)$ as in 6.3.1. For every node i of T let $C(i) \ni i$ be the set of descendants of i, and $G(i) = G[\bigcup_{j \in C(i)} b(j)]$.

To count the number of q-colourings of G we compute, for every node i of the tree T and every mapping $c : b(i) \to [q]$, the number $d(i, c)$ of q-colourings of $G(i)$ that extend c. Especially, $d(i, c) = 0$ if c is not a proper colouring of $G[b(i)]$. For every $q \geq 0$, the empty graph $(\emptyset, \emptyset)$ has exactly one q-colouring which we call $c_\varnothing$. For the root r of T we have $\#\mathsf{col}_q(G) = d(r, c_\varnothing)$. The values $d(i, c)$ are computed bottom up from the leaves of T to its root r as follows.

leaf For every leaf l of T we have $a(l, c_\varnothing) = 1$, because $c_\varnothing$ is the only q-colouring of the empty graph $(\emptyset, \emptyset)$.

introduce Let i be an introduce node with child j and let $c : b(i) \to [q]$. If c is not a proper colouring of $G[b(i)]$ then $d(i, c) = 0$. Otherwise, $d(i, c) = d(j, c')$ where c' is the restriction of c to $b(j)$. The latter holds true because, in $G(i)$, the new vertex $w \in b(i) \setminus b(j)$ has all its neighbours in $b(i)$.

forget Let i be an forget node with child j and let $c : b(i) \to [q]$. We have $d(i, c) = \sum_{c'} d(j, c')$, where the sum is taken over all $c' : b(j) \to [q]$ that extend c.

join Let i be a join node with children j and j' and let $c : b(i) \to [q]$. We have $d(i, c) = d(j, c) \cdot d(j', c)$, because there are no edges between $C(j) \setminus b(i)$ and $C(j') \setminus b(i)$ in G.

root G has $\#\mathsf{col}_q(G) = d(r, c_\varnothing)$ proper q-colourings because they all extend $c_\varnothing : b(r) \to [q]$.

The decomposition tree T of G has size $\mathcal{O}(n)$, and we have shown how to construct a T of width less than $4b$. Then, for each bag we have at most q^{4b} colourings to consider. Each of them needs $\mathcal{O}(b \log q)$ time to look up the information from its children. Thus the overall running time, including the time to find the tree decomposition is $\mathcal{O}(bq^{4b} n \log q + b^2 n \log n)$. Thus exactly counting q-colourings when the parameter is maximum vertex size is in FPT for thick forests. This is polynomial if $b = \mathcal{O}(\log n / \log q)$.

Remark 13. The algorithm correctly counts q-colourings in any graph of bounded treewidth, not just thick trees. Constructing the tree decomposition is in FPT, but not in polynomial time. See Remark 12. Thus counting q-colourings for all graphs with treewidth as parameter is is in FPT.

Acknowledgment

We thank Mark Jerrum for input and useful comments on earlier versions of this paper.

References

[1] G. Agnarsson (2003), On chordal graphs and their chromatic polynomials, *Mathematica Scandinavica* **93**, 240–246.

[2] S. Arnborg, D. G. Corneil and A. Proskurowski (1987), Complexity of finding embeddings in a k-tree, *SIAM Journal on Algebraic Discrete Methods* **8**, 277–284.

[3] M. Albertson, R. Jamison, S. Hedetniemi and S. Locke (1989), The subchromatic number of a graph, *Discrete Mathematics* **74**, 33–49.

[4] A. Berry and R. Pogorelcnik, A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph, *Information Processing Letters* **111**, 508–511.

[5] V. Boncompagni, I. Penev and K. Vuskovic (2019), Clique-cutsets beyond chordal graphs, *Journal of Graph Theory* **91**, 192-246.

[6] D. Bovet and P. Crescenzi (2006), Introduction to the theory of complexity, https://www.pilucrescenzi.it/wp/wp-content/uploads/2017/09/itc.pdf.

[7] M. Belbasi and M. Fürer (2022), An improvement of Reed's treewidth approximation, arXiv:2010.03105.

[8] H. L. Bodlaender (1993), A tourist guide through treewidth. *Acta Cybernetica* vol. **11**, 1–21.

[9] H. L. Bodlaender (1996), A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* vol. **25** 1305–1317.

[10] H. L. Bodlaender (1998), A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* vol. **209**, 1–45.

[11] H. Broersma, F. Fomin, J.Nešetřil and G. Woeginger (2002), More about subcolorings, *Computing* **69**, 187–203.

[12] J. Brown (1996), The complexity of generalized graph colorings, *Discrete Applied Mathematics* **69**, 257-270.

[13] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas (2006), The strong perfect graph theorem, *Annals of Mathematics* **164**, 51–229.

[14] M. Chudnovsky, A. Scott, P. Seymour and S. Spirkl (2020), Detecting an odd hole, *Journal of the ACM* **67**, Article 5, 1–12.

[15] A. Ćustić, B. Klinz and G. Woeginger (2015), Geometric versions of the three-dimensional assignment problem under general norms, *Discrete Optimization* **18**, 38–55.

[16] D. Cohen and P. Jeavons (2006), The complexity of constraint languages, in *Handbook of Constraint Programming*, Elsevier, pp. 169–204.

[17] G. Cornuejols, X. Liu and K. Vušković (2003), A polynomial algorithm for recognizing perfect graphs, in *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS2003)*, pp. 20-27.

[18] B. Courcelle (1992), The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues, *RAIRO. Informatique Théorique et Applications* **26**, 257–286.

[19] R. Diestel, *Graph Theory*, 4th edition, Graduate Texts in Mathematics **173**, Springer, 2012.

[20] M. Dyer, L.A. Goldberg, C. Greenhill and M. Jerrum (2004), The relative complexity of approximate counting problems, *Algorithmica* **38**, 471–500.

[21] M. Dyer and C. Greenhill, The complexity of counting graph homomorphisms, *Random Structures & Algorithms* **17** (2000), 260–289.

[22] M. Dyer, M. Jerrum, H. Müller and K. Vušković (2021), Counting weighted independent sets beyond the permanent, *SIAM Journal on Discrete Mathematics* **35**, 1503–1524.

[23] T. Ekim, P. Hell, J. Stacho and D. de Werra (2008), Polarity of chordal graphs, *Discrete Applied Mathematics* **156**, 2469–2479.

[24] E. Eschen and X. Wang (2014), Algorithms for unipolar and generalized split graphs, *Discrete Applied Mathematics* **162**, 195–201.

[25] E. Eschen, C. Hoàng, M. Petrick and R. Sritharan (2005), Disjoint clique cutsets in graphs without long holes, *Journal of Graph Theory* **48**, 277–298.

[26] M. Fellows (2002), Parameterized complexity: the main ideas and connections to practical computing, *Electronic Notes in Theoretical Computer Science* **61**, 1–19.

[27] J. Fiala, K. Jansen, V.B. Le and E. Seidel (2003), Graph subcolorings: complexity and algorithms, *SIAM Journal on Discrete Mathematics* **16**, 635—650.

[28] A. Farrugia (2004), Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard, *Electronic J. Combinatorics* **11**, Research Paper #46, 9 pp.

[29] T. Feder and P. Hell (2006), Matrix partitions of perfect graphs, *Discrete Mathematics* **306**, 2450–2460.

[30] M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, 1988.

[31] F. Gavril (1974), The intersection graphs of subtrees in trees are exactly the chordal graphs, *Journal of Combinatorial Theory Series B* **16**, 47–56.

[32] R. Hayward (1985), Weakly triangulated graphs, *Journal of Combinatorial Theory B* **39**, 200–209.

[33] P. Heggernes (2006), Minimal triangulations of graphs: A survey, *Discrete Mathematics* **306**, 297–317.

[34] P. Heggernes and D. Kratsch (2007), Linear-time certifying recognition algorithms and forbidden induced subgraphs, *Nordic Journal of Computing* **14**, 87–108.

[35] P. Jégou (1993), Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems, in *Proc. AAAI-93*, AAAI Press, pp. 731-736.

[36] M. Jerrum, A. Sinclair and E. Vigoda (2004), A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries, *Journal of the ACM* **51**, 671–697.

[37] M. Jerrum, *Counting, sampling and integrating: algorithms and complexity*, Lectures in Mathematics – ETH Zürich, Birkhäuser, 2003.

[38] M. Jerrum, L. Valiant and V. Vazirani (1986), Random generation of combinatorial structures from a uniform distribution, *Theoretical Computer Science* **43**, 169–188.

[39] I. Kanj, C. Komusiewicz, M. Sorge and E. van Leeuwen (2018), Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs, *Journal of Computer and System Sciences* **92**, 22–47.

[40] T. Kloks, Treewidth – computations and approximations. Springer-Verlag Berlin, *Lecture Notes in Computer Science* vol. **842**, 1994.

[41] B. Lévêque, F. Maffray, B. Reed and N. Trotignon (2009), Coloring Artemis graphs, *Theoretical Computer Science* **410**, 2234-2240.

[42] L. Lovász (1972), Normal hypergraphs and the perfect graph conjecture, *Discrete Mathematics* **2**, 253–267.

[43] G. MacGillivray and M.L. Yu (1999), Generalized partitions of graphs, *Discrete Applied Mathematics* **91**, 143–153.

[44] C. McDiarmid and N. Yolov (2016), Recognition of unipolar and generalised split graphs, *Algorithms* **8**, 46–59.

[45] S. Nikolopoulos and L. Palios (2007), Hole and antihole detection in graphs, *Algorithmica* **47**, 119-138.

[46] D. Rose, R. Tarjan and G. Lueker (1976), Algorithmic aspects of vertex elimination on graphs, *SIAM Journal on Computing* **5**, 266–283.

[47] S. Rihane, C. Adegbindin and A. Togbé (2020), Fermat Padovan and Perrin numbers, *Journal of Integer Sequences* **23**, Article 20.6.2.

[48] A. Salamon and P. Jeavons (2008), Perfect constraints are tractable, in *Proc. 14th International Conference on Principles and Practice of Constraint Programming (CP 2008)*, LNCS **5202**, pp. 524–528.

[49] J. Stacho (2008), On 2-subcolourings of chordal graphs, in: *LATIN 2008 : Theoretical Informatics*, Springer LNCS **4957**, pp. 544–554.

[50] J. Stacho, *Complexity of generalized colourings of chordal graphs*, Ph.D. thesis, Simon Fraser University, 2008.

[51] R. Tarjan (1985), Decomposition by clique separators, *Discrete Mathematics* **55**, 221–232.

[52] R. Tyshkevich and A. Chernyak (1985), Algorithms for the canonical decomposition of a graph and recognizing polarity, Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk 6, 16–23. (in Russian)

[53] L. Valiant (1979), The complexity of computing the permanent, *Theoretical Computer Science* **8**, 189–201.

[54] M. Yannakakis (1981), Computing the minimum fill-in is NP-complete, *SIAM J. Alg. Disc. Meth.* **2**, 77–79.