H-colouring P_t-free graphs in subexponential time

Carla Groenland[1], Karolina Okrasa[2], Paweł Rzążewski[2], Alex Scott*[1], Paul Seymour†[3], and Sophie Spirkl[3]

[1]University of Oxford, {groenland,scott}@maths.ox.ac.uk
[2]Faculty of Mathematics and Information Science, Warsaw University of Technology,
okrasak@student.mini.pw.edu.pl,p.rzazewski@mini.pw.edu.pl
[3]Princeton University, {pds,sspirkl}@math.princeton.edu

March 29, 2019

Abstract

A graph is called P_t-free if it does not contain the path on t vertices as an induced subgraph. Let H be a multigraph with the property that any two distinct vertices share at most one common neighbour. We show that the generating function for (list) graph homomorphisms from G to H can be calculated in subexponential time $2^{O\left(\sqrt{tn\log(n)}\right)}$ for $n = |V(G)|$ in the class of P_t-free graphs G. As a corollary, we show that the number of 3-colourings of a P_t-free graph G can be found in subexponential time. On the other hand, no subexponential time algorithm exists for 4-colourability of P_t-free graphs assuming the Exponential Time Hypothesis. Along the way, we prove that P_t-free graphs have pathwidth that is linear in their maximum degree.

Keywords: colouring, P_t-free, subexponential-time algorithm, partition function, path-decomposition.

1 Introduction

Throughout this paper, graphs do not have multiple edges or loops. When we need general graphs (with multiple edges and loops), we call them multigraphs. We use the notation vv' for the edge $\{v, v'\}$. For a multigraph G, the set $N_G(v) = \{v' : vv' \in E(G)\}$ contains v if and only if G has a loop at vertex v.

*Supported by a Leverhulme Trust Research Fellowship.
†Supported by ONR grant N00014-14-1-0084 and NSF grant DMS-1265563.

A *k-colouring* of a graph G is a function $c : V(G) \to \{1, \dots, k\}$ such that $c(v) \neq c(v')$ for all $vv' \in E(G)$. The decision problem k-COLOURABILITY asks whether a given graph G is k-colourable. This problem is NP-complete for $k \geqslant 3$ in general. In order to investigate what graph structure makes the decision problem hard, a natural question is whether the problem becomes easy if it is restricted to instances that do not contain a particular structure. Thus we restrict to the class of F-free graphs, that is, those graphs which do not contain F as an induced subgraph, for some fixed graph F. If k is part of the input, a full classification is given by Král et al. [18]. For fixed $k \geqslant 3$, k-COLOURABILITY is shown to be NP-complete for F-free graphs if F is either a cycle C_ℓ for $\ell \geqslant 3$ [17] or the claw $K_{1,3}$ [14, 19]. Any graph F which does not contain a cycle nor the claw is a disjoint union of paths.

Let P_t denote the path on t vertices. Polynomial-time algorithms for deciding k-COLOURABILITY for P_t-free graphs exist for $t \leqslant 5$ [13], $(k, t) = (4, 6)$ [6, 7] and $(k, t) = (3, 7)$ [4]. On the other hand, Huang [15] showed 4-COLOURABILITY is NP-complete for P_7-free graphs and 5-COLOURABILITY is NP-complete for P_6-free graphs. It is an open problem to determine the complexity of 3-COLOURABILITY of P_t-free graphs for $t \geqslant 8$.

It is also open whether MAXIMUM INDEPENDENT SET is decidable in polynomial time for P_t-free graphs for $t \geqslant 7$. Brause [5] and Bascó, Marx and Tuza [2] independently showed a greedy exhaustive approach yields a subexponential-time algorithm for MAXIMUM INDEPENDENT SET on P_t-free graphs. In this paper we show that there are subexponential-time algorithms for a larger class of problems, including 3-COLOURABILITY and MAXIMUM INDEPENDENT SET, and also give counting results. Our algorithm builds on the following property of P_t-free graphs.

Lemma 1. *A P_t-free graph of maximum degree Δ has pathwidth at most $(\Delta-1)(t-2)+1$. Moreover, a path-decomposition of this width can be found in polynomial time.*

This lemma is an improvement on the treewidth bound of Bascó et al. [1] for P_t-free graphs (which they used to improve the algorithm of Bascó et al. [2]). The result is tight up to a constant factor, which can be seen by replacing the vertices of a binary tree by cliques.

We now introduce our framework. Let H be a multigraph and G a (simple) graph. A *graph homomorphism* from G to H is a map $f : V(G) \to V(H)$ such that $vv' \in E(G)$ implies $f(v)f(v') \in E(H)$. (Thus a 3-colouring is a graph homomorphism to K_3.) A *list H-colouring instance* $I = (G, L)$ consists of a graph G together with a function $L : V(G) \to \mathcal{P}(V(H))$ that assigns a subset $L_v \subseteq V(H)$ to every $v \in V(G)$. A *list H-colouring* of such an instance is a graph homomorphism f from G to H such that $f(v) \in L_v$ for all $v \in V(G)$. We denote the set of list H-colourings of (G, L) by $\mathcal{LC}((G, L), H)$.

A useful way to summarise information about H-colourings of a graph G is to use a multivariate generating function (see for example [21]). Given a multigraph

H, we define the *partition function* $p_{(G,L)\to H}(x)$ by

$$p_{(G,L)\to H}(x) := \sum_{f\in\mathcal{LC}((G,L),H)} \prod_{u\in V(G)} w_{u,f(u)} x_{f(u)}.$$

We omit the lists L where clear from context. The weights $w_{v,h}$ (for $v \in V(G)$ and $h \in V(H)$) are included to allow more general application and can be ignored by choosing them identically one. For $H = (\{h,h'\},\{hh',h'h'\})$, $p_{G\to H}(x)$ gives the independent set polynomial when $x_{h'}$ is set to one.

Summing appropriate coefficients of the polynomial, the partition function can be used to for example count the number of list H-colourings, or to count the number of "restrictive H-colourings" [8] in which a restriction is placed on the size of the preimages of the vertices of H.

The following theorem is our main result and will be proved in Section 3.

Theorem 2. *Let H be a multigraph so that $|N_H(h) \cap N_H(h')| \leqslant 1$ for all distinct vertices h, h' of H. For $t \geqslant 4$, the polynomial $p_{G\to H}(x)$ can be calculated for every P_t-free graph G in time $2^{O\left(\sqrt{tn\log(n)}\right)}$ where $n = |V(G)|$.*

For simple graphs H, the condition $|N_H(h) \cap N_H(h')| \leqslant 1$ for all distinct h, h' is equivalent to H not having C_4 as (not necessarily induced) subgraph.

Corollary 3. *The following problems can be solved for P_t-free graphs G in time $2^{O\left(\sqrt{tn\log(n)}\right)}$ where $n = |V(G)|$:*

- *Counting the number of H-colourings for any fixed simple graph H with no C_4 subgraph.*
- *Computing the independent set polynomial.*

In particular, it can be decided in subexponential time whether a P_t-free graph is 3-colourable (and the number of 3-colourings can be counted).

For example, if G is P_t-free and H is an odd cycle, then we can count the number of H-colourings of G in subexponential time. This problem is $\#P$-complete if all graphs G are allowed [9] and the corresponding decision problem is NP-complete [12].

The Exponential Time Hypothesis (ETH) [16] states that there is an $\epsilon > 0$ such that there is no $O(2^{\epsilon n})$-time algorithm for 3-SAT. In Section 4 we prove the following result that shows that it is in some sense unlikely that Corollary 3 will extend to k-colouring P_t-free graphs for $k > 3$.

Proposition 4. *If the Exponential Time Hypothesis is true, then there is no algorithm running in time subexponential in the number of vertices of the graph for*

- 4-COLOURABILITY *on P_7-free graphs;*
- 3-COLOURABILITY *for F-free graphs for any connected F which is not a path.*

This result might shed some light on why the complexity status of 3-COLOURABILITY of P_t-free graphs for t large has remained open whereas the complexity of k-COLOURABILITY of F-free graphs has been settled for other values of k or F.

2 Pathwidth of P_t-free graphs and dynamic programming

A *path-decomposition* of a graph G is a sequence of subsets X_i of vertices of G with three properties:

- The vertex set of G equals $\bigcup_i X_i$,
- For each edge of G, there exists an i such that both endpoints of the edge belong to subset X_i, and
- $X_\ell \cap X_j \subseteq X_i$ for every three indices $1 \leqslant \ell \leqslant i \leqslant j$.

The *pathwidth* of G is defined as the minimum of $\max_i |X_i| - 1$ over the path-decompositions of G.

Let T be a rooted tree with root r. For a vertex v, let T_v denote the path of T between r and v. For each vertex w, fix a linear order of the set of children of w. We call this a *plane tree*. If u, v are both children of w and u is earlier than v in the corresponding ordering, we say u is an *elder sibling* of v.

Let T be a spanning tree of a graph G, equipped with orders to make it a plane tree. We call it an *uncle tree* of G if for every edge uv of G that is not an edge of T, one of u, v has an elder sibling that is an ancestor of the other. We use the following result of Seymour [22].

Theorem 5. *For every connected graph we can compute an uncle tree with any specified vertex as root in polynomial time.*

(The proof is easy; grow a depth-first tree, subject to the condition that the path of the tree between each vertex and the root is induced.)

Proof of Lemma 1. We may assume that G is connected. Take an uncle tree T of G, and order its leaves as $p_1, \ldots, p_k$ say, in the natural order of the leaves of a plane tree. For $1 \leqslant i \leqslant k$, let X_i be the set of vertices of T that either belong to T_{p_i}, or have an elder sibling (and hence also a parent) in this set. We claim that the sequence $(X_1, \ldots, X_k)$ is a path-decomposition of G. We check that:

- Every vertex belongs to some X_i (this is clear).
- For all u, v adjacent in G, there exists i with $u, v \in X_i$. To see this, since T is an uncle tree we may assume that u has an elder sibling u' that is an ancestor of v. Choose a leaf p_i of T such that T_{p_i} contains v; then the common parent of u, u' belongs to T_{p_i}, and hence $u, v \in X_i$.
- $X_\ell \cap X_j \subseteq X_i$ for $1 \leqslant \ell \leqslant i \leqslant j \leqslant k$. To see this, let $v \in X_\ell \cap X_j$; consequently either v or an elder sibling of v belongs to T_{p_ℓ}, and either v or an elder sibling belongs to T_{p_j}. It follows that either v or an elder sibling of v belongs to T_{p_i}, from the ordering of the leaves of T.

This proves the claim.

If T is an uncle tree of G, then each path of T starting from r is induced in G; and so if G does not contain P_t, then $|T_{p_i}| \leqslant t-1$ for each i, and so $|X_i| \leqslant \Delta - 1 + (t-3)(\Delta-2)+1$, where Δ denotes the maximum degree of G. $\square$

We give a short outline of how the standard dynamic programming approach can be applied to compute $p_{G\to H}(x)$ in time $2^{O(p)}n$ given a path-decomposition of width p of a graph G on n vertices.

Let $(X_1,\dots,X_k)$ be the given path-decomposition of G. We define $X(i) = \bigcup_{j\leqslant i} X_j$. For each $i \in [k]$ and list H-colouring $g : X_i \to V(H)$, we compute the polynomial $p_{G[X(i)]\to H}$ with the vertices in X_i precoloured: we define $p_i(g)$ as the sum, over the list H-colourings $f : X(i) \to V(H)$ such that $f|_{X_i} = g$, of

$$\prod_{u\in X(i)} w_{u,f(u)} x_{f(u)}.$$

For each list H-colouring $g : X_1 \to V(H)$, we set $p_1(g) = \prod_{v\in X_1} w_{v,g(v)} x_{g(v)}$. Having computed all $p_i(g)$ for some i, for each list H-colouring $g : X_{i+1} \to V(H)$ we select the list H-colourings $g_1,\dots,g_\ell$ of X_i that are compatible with g, that is, $g_i(v) = g(v)$ for $v \in X_i \cap X_{i+1}$ and $g(u)g(v) \in E(H)$ if $uv \in E(G)$ for all $u \in X_i$ and $v \in X_{i+1}$. Since $N_G[v] \cap X(i) \subseteq X_i$ for all $v \in X_{i+1} \setminus X_i$, we can then compute

$$p_{i+1}(g) = \sum_{j=1}^{\ell} p_i(g_j) \prod_{v\in X_{i+1}\setminus X_i} w_{v,g(v)} x_{g(v)}.$$

Finally, we calculate the desired $p_{G\to H}$, which is the sum, over all list H-colourings g of X_k, of $p_k(g)$.

3 Algorithm and time analysis

Throughout this section, H is a fixed multigraph such that $|N_H(h) \cap N_H(h')| \leqslant 1$ for all distinct h, h' in H. We allow loops in H but no multiple edges.[1] The P_t-free graphs G are assumed to be simple.

We shall say a list colouring instance $I = (G, L)$ has *weight* $w(I) = \sum_{v\in V(G)} |L_v|$ and is *reduced* if $|L_v| \geqslant 2$ for all $v \in V(G)$. The key observation we need is the following.

Lemma 6. *Let $I = (G, L)$ be a reduced list H-colouring instance and let $v \in V(G)$ with degree $d(v)$. For $h \in V(H)$, let*

$$C_h = \{v' \in N_G(v) : L_{v'} \subseteq N_H(h)\}.$$

Then there is at most one $h \in L_v$ for which $|C_h| > \frac{1}{2}d(v)$.

[1] It is possible to extend the algorithm to compute a version of $p_{G\to H}(x)$ with edge weights $A_{h,h'}$ for $h, h' \in V(H)$, but in this case the weights $w_{v',h'}$ have to be updated in Line 3 and 6 of Algorithm HCol to $w_{v',h'}A_{h,h'}$ for all $v' \in N_v$.

Proof. Suppose $h \neq h'$ in L_v both satisfy $|C_h|, |C_{h'}| > \frac{1}{2}d(v)$. Then there exists $v' \in N_G(v)$ such that $v' \in C_h \cap C_{h'}$. Hence $L_{v'} \subseteq N_H(h) \cap N_H(h')$, so that by our assumption on H we find $|L_{v'}| \leqslant 1$, contradicting the assumption that I is reduced. □

This lemma tells us that "colouring" a vertex v of degree $d(v) = \Delta$ decreases the weight of a reduced instance by at least $\frac{1}{2}\Delta$ for all but one "colour" in L_v. Either there is a vertex of high degree and we can reduce the weight significantly by colouring this vertex, or Δ is "small" and we can apply the results from the previous section to compute $p_{G\to H}(x)$ in time $2^{O(t\Delta)}$.

Our algorithm "HCol" for computing the list H-colouring function of a graph G is given below. This algorithm either terminates or recurses on instances of strictly smaller weight. Therefore, it always terminates in finite time. We can represent the recursions by a tree: the root is the first call of the algorithm and each recursive call creates a child. For P_t-free graphs, we can bound the number of nodes in this recursion tree.

Proposition 7. *Let $t \geqslant 4$, $c > 4\sqrt{t|V(H)|/\log(2)}$ and $f(w) = 2^{c\sqrt{w\log(w)}}$. Then there exists an n_0 for Algorithm HCol such that if it is applied to an instance $I = (G, L)$ of weight $w(I)$ with G a P_t-free graph, then the number of nodes in the corresponding recursion tree is bounded by $f(w(I))$.*

Algorithm HCol: Outputs the list H-colouring function.

Input: a list H-colouring instance $I = (G, L)$ for $G = (V, E)$.
1. If $|V| \leqslant n_0$, compute the list H-colouring function exhaustively.
2. If there exists $v \in V$ such that $|L_v| = 0$, return 0.
3. If there exists $v \in V$ such that $|L_v| = 1$, say $L_v = \{h\}$, then set $L'_{v'} = L_{v'} \cap N_H(h)$ for $v' \in N_G(v)$ and $L'_{v'} = L_{v'}$ for $v' \notin N_G(v)$. Return $w_{v,h}x_h\mathrm{HCol}(G - v, L')$.
4. If G is not connected, let $G_1, \ldots, G_k$ be the connected components. Return $\prod_{i=1}^{k}\mathrm{HCol}(G_i, L|_{V(G_i)})$.
5. If the maximum degree of G is at most $\sqrt{n\log(n)/t}$, compute a path-decomposition of G of width $O\left(\sqrt{tn\log(n)}\right)$ and compute the result using dynamic programming.
6. Otherwise take $v \in V$ of maximal degree. For $h \in L_v$, set $L^h_{v'} = L_v \cap N_H(h)$ if $v' \in N_G(v)$ and $L^h_{v'} = L_{v'}$ if $v' \notin N_G(v)$. Return $\sum_{h\in L_v} w_{v,h}x_h \; \mathrm{HCol}(G - v, L^h)$.

Algorithm HCol gives the correct answer for any graph G: in line 2 we note that if some vertex has an empty list, then $p_{G\to H} = 0$; in line 3 we note that if the list of a vertex $v \in V(G)$ has a single element $h \in V(H)$, then v has to be mapped to h, i.e. $p_{G\to H}(x) = w_{v,h}x_h p_{G-v\to H}(x)$; in line 4 we use the algebraic identity $p_{G_1\sqcup G_2\to H}(x) = p_{G_1\to H}(x)p_{G_2\to H}(x)$; in line 6, we use $p_{G\to H}(x) = \sum_{h\in L_v} w_{v,h}x_h p_{G-v\to H}(x)$.[2]

[2]We left the lists of the vertices implicit in the notation; the precise way in which the lists need to be updated is given in the algorithm.

Inspecting and updating the lists of vertices, finding a vertex of maximal degree and finding the connected components of a graph on n vertices can all be done in time Cn^2, where the constant C may depend on H and n_0. Line 5 is applied at most once per node and takes $2^{O\left(\sqrt{tn\log(n)}\right)}$. Since $|L_v| \leqslant |V(H)|$ for all $v \in V(G)$, it follows that $w(I) \leqslant |V(H)|n = O(n)$. Theorem 2 follows from Proposition 7 by observing that $Cn^2 2^{O\left(\sqrt{tn\log(n)}\right)} 2^{O\left(\sqrt{tn\log(n)}\right)} = 2^{O\left(\sqrt{tn\log(n)}\right)}$.

We require the following simple estimate.

Lemma 8. *Let $m, y > 0$ and $c > y^{-1}$. There exists an $n_0 \in \mathbb{N}$ such that $f(w) = e^{c\sqrt{w\log(w)}}$ satisfies*

$$f(n-2) + mf\left(n - y\sqrt{n\log(n)}\right) \leqslant f(n)$$

for all $n \geqslant n_0$.

Proof. Let $\epsilon > 0$ be given such that $\sqrt{1-x} \leqslant 1 - \frac{1}{2}x$ and $e^{-x} \leqslant 1 - x/2$ for all $0 \leqslant x \leqslant \epsilon$. Choose n_0 sufficiently large such that $2/n, y\sqrt{\log(n)/n}, c\log(n)/\sqrt{n} \leqslant \epsilon$ and $mn^{-cy/2} < \frac{c}{2}\sqrt{\log(n)/n}$ for all $n \geqslant n_0$ (by assumption, $cy > 1$). We calculate

$$\begin{aligned} f(n-2) + mf\left(n - y\sqrt{n\log(n)}\right) &\leqslant e^{c\sqrt{(n-2)\log(n)}} + me^{c\sqrt{\log(n)}\left(n - y\sqrt{n\log(n)}\right)^{\frac{1}{2}}} \\ &= f(n)^{\sqrt{1-2/n}} + mf(n)^{\left(1-y\sqrt{\log(n)/n}\right)^{\frac{1}{2}}} \\ &\leqslant f(n)^{1-1/n} + mf(n)^{1-\frac{1}{2}y\sqrt{\log(n)/n}} \\ &= f(n)\left[e^{-c\sqrt{\log(n)/n}} + me^{-\frac{1}{2}cy\log(n)}\right] \end{aligned}$$

But

$$e^{-c\sqrt{\log(n)/n}} + me^{-\frac{1}{2}cy\log(n)} \leqslant 1 - \frac{1}{2}c\sqrt{\log(n)/n} + mn^{-\frac{1}{2}cy} < 1. \qquad \square$$

Proof of Proposition 7. Let n_0 be given from Lemma 8 applied with $m = |V(H)|$, $y = \frac{1}{4\sqrt{t|V(H)|}}$ and using $c\log(2)$ instead of c. Enlarging n_0 if necessary for the last three properties, we may now assume that

$$\begin{aligned} f(w-2) + |V(H)|f\left(w - y\sqrt{w\log(w)}\right) &\leqslant f(w) && \text{for all } w \geqslant n_0, \\ f(k) + f(\ell) + 1 &\leqslant f(k+\ell) && \text{for all } k, \ell \geqslant n_0, \\ f(k) + (w-k) + 1 &\leqslant f(w) && \text{for all } w \geqslant n_0,\ k < w, \\ w + 1 &\leqslant f(w) && \text{for all } w \geqslant n_0. \end{aligned}$$

The proposition is proved by induction on $w = w(I)$. If the algorithm terminates in line 1, 2 or 5, then there is only one iteration and $f(n) \geqslant 1$ for all $n \in \mathbb{Z}_{\geqslant 0}$.

If the algorithm reaches line 3, then G has at least n_0 vertices and $|L_v| \geqslant 1$ for all $v \in V$, so that $w(I) \geqslant n_0$. Therefore, the statement holds for all $w < n_0$.

Suppose the statement has been shown for instances with $w(I) < w$ for some $w \geqslant n_0$. If the algorithm recurses on line 3, then the removed vertex contributed at least 1 to the weight, and so by induction at most $f(w-1)+1 \leqslant f(w)$ iterations are taken.

If the algorithm reaches line 4, we may assume the instance is reduced, and so $w(I) \leqslant 2|V(G)|$. Suppose the graph G is disconnected with connected components $G_1, \ldots, G_k$. Let $I_i = (G_i, L|_{V(G_i)})$ and note that I_i is also reduced. Hence if $|V(G_i)| \leqslant \frac{1}{2}w(I_i) \leqslant n_0$, then the recursive call on G_i will take a single iteration. Renumber so that $I_1, \ldots, I_\ell$ have weight at most $2n_0$ and $I_{\ell+1}, \ldots, I_k$ have weight at least $2n_0$. By induction the algorithm takes at most

$$\sum_{i=1}^{\ell} 1 + \sum_{i=\ell+1}^{k} f(w(I_i)) + 1 \leqslant f(w)$$

iterations, where the inequality follows from the assumptions we placed on n_0, considering $k-\ell=0$, $k-\ell=1$ and $k-\ell>1$ separately.

At line 6, the hypothesis of Lemma 6 is satisfied. All but one of the instances have their weight reduced by at least

$$\frac{1}{2}\sqrt{n\log(n)/t} \geqslant \frac{1}{2\sqrt{t|V(H)|}}\sqrt{w\left(\log(w) - \log(|V(H)|)\right)} \geqslant y\sqrt{w\log(w)}$$

(using that $w = \sum_{v\in V(G)} |L_v| \leqslant |V(H)|n$ and assuming $\log(w) - \log(|V(H)|) \geqslant \frac{1}{4}\log(w)$ by enlarging n_0 if necessary). The other instance has its weight reduced by at least 2, since the vertex v with $|L_v| \geqslant 2$ is removed. By induction, the number of nodes in the recursion tree is at most $f(w-2)+(|V(H)|-1)f\left(w - y\sqrt{w\log(w)}\right) + 1 \leqslant f(w)$. □

4 Extensions

Our algorithm can easily be adapted to find the H-colouring f of G which minimises the cost $\sum_{v\in V(G)} w_{v,f(v)}$; in particular, Minimum Cost Homomorphism [11] can be solved in subexponential time for G and H as above.

Note that Lemma 6 is the bottleneck for extending the time complexity to, for example, 4-colouring ($H = K_4$): it is possible that the neighbourhood of a vertex v with $L_v = \{1,2,3,4\}$ has mostly neighbours with list $\{1,2\}$, so that both C_3 and C_4 are large. Assuming the Exponential Time Hypothesis (ETH), this is to be expected in view of Proposition 4: under ETH, no such subexponential time algorithm can exist for 4-colourability on P_7-free graphs.

Proof of Proposition 4. Huang [15] gives a reduction of an instance of 3-sat with m formulas and n variables into an instance of 4-colourability for a P_7-free graph on $O(n+m)$ vertices. Therefore, any algorithm for 4-colourability on

P_7-free graphs yields an algorithm for 3-SAT with the same time dependence on the input size.

Let F be a connected graph which is not a path. Then F contains either the claw $K_{1,3}$ or a cycle. We prove ETH implies that there is no subexponential time algorithm for 3-COLOURABILITY of F-free graphs. The standard reduction (for example [10, Prop. 2.26]) from 3-SAT to 3-COLOURABILITY creates a graph on $O(n + m)$ vertices. Kamiński and Lozin [17] reduce 3-COLOURABILITY to 3-COLOURABILITY on graphs of girth at least g (for every $g \geqslant 3$) by (in the worst case) replacing each vertex by a constant-sized gadget. This handles the case when F contains a cycle.

For F containing the claw $K_{1,3}$, note that claw-free graphs are a superset of line graphs. Holyer [14] reduces 3-SAT to 3-EDGE COLOURABILITY on 3-regular graphs. Given n variables and m clauses, constant-sized gadgets are created for each variable and clause; additional components are added to the variable gadgets for each time it occurs in a clause. Since there are at most $3m$ such occurrences, this creates a graph on $O(n+m)$ vertices. Since the graph is 3-regular, the number of edges is $O(n + m)$ as well. Hence the line graph of such a graph will have $O(n + m)$ vertices. Since 3-colourings of the vertices of the line graph are in one-to-one correspondence with 3-colourings of the edges of the original graph, we can hence reduce 3-SAT to 3-COLOURABILITY of line graphs on $O(n+m)$ vertices. □

If ETH holds, then any polynomial-time reduction from 3-SAT to a problem with a subexponential algorithm must "blow up" the instance size. Our result therefore suggests that, if one attempts to prove NP-completeness of 3-COLOURABILITY for P_t-free graphs (for t large) by designing gadgets, it may be necessary either to start from a problem whose instance size has already been "blown up" or to use gadgets which are not of bounded size.

Our algorithm only uses the property of P_t-free graphs that every induced subgraph has pathwidth $O(t\Delta)$. Seymour [22] proves that a tree-decomposition of width $O(\ell\Delta)$ can be computed efficiently for graphs that do not contain cycles of length at least ℓ as induced subgraph; therefore, our algorithm extends to this class of graphs (after adjusting the standard dynamic programming approach of for example [3] to our setting in a similar fashion as done in Section 2). This motivates the following question.

Problem. *For fixed t, are* 3-COLOURABILITY *and* MAXIMUM INDEPENDENT SET *solvable in polynomial time on graphs that have no induced cycles of length greater than t?*

Theorem 5 can also be used to bound tree-depth of P_t-free graphs. The *tree-depth* of a graph G is the minimum height of a forest F on the same vertex set with the property that for every edge of G, the corresponding vertices are in an ancestor-descendant relationship to each other in F [20].

Corollary 9. *The tree-depth of a connected P_t-free graph G of maximum degree Δ is at most $(t-2)(\Delta-1)+1$.*

Such a desired forest F can be computed as follows. First compute an uncle tree T for G. For each non-leaf vertex $v \in T$, create a path P_v containing all the children of v. The forest F is obtained by connecting the "end"point of a path P_v to the "start"points of the paths of its children. Now recall that an uncle tree has height at most $t-1$ since each path from the root to a leaf is induced and that a non-root, non-leaf node has at most $\Delta - 1$ children.

References

[1] Bacsó, G., Lokshtanov, D., Marx, D., Pilipczuk, M., Tuza, Z., and van Leeuwen, E. J. Subexponential-time algorithms for maximum independent set in P_t-free and broom-free graphs. *Algorithmica 81*, 2 (2019), 421–438.

[2] Bacsó, G., Marx, D., and Tuza, Z. H-free graphs, independent sets, and subexponential-time algorithms. In *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)* (2017), J. Guo and D. Hermelin, Eds., vol. 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 3:1–3:12.

[3] Bodlaender, H. L., Bonsma, P., and Lokshtanov, D. The fine details of fast dynamic programming over tree decompositions. In *Parameterized and Exact Computation* (Cham, 2013), G. Gutin and S. Szeider, Eds., Springer International Publishing, pp. 41–53.

[4] Bonomo, F., Chudnovsky, M., Maceli, P., Schaudt, O., Stein, M., and Zhong, M. Three-coloring and list three-coloring of graphs without induced paths on seven vertices. *Combinatorica* (2017).

[5] Brause, C. A subexponential-time algorithm for the maximum independent set problem in P_t-free graphs. *Discrete Applied Mathematics 231* (2017), 113–118.

[6] Chudnovsky, M., Spirkl, S., and Zhong, M. Four-coloring P_6-free graphs. I. Extending an excellent precoloring. *arXiv:1802.02282* (2018).

[7] Chudnovsky, M., Spirkl, S., and Zhong, M. Four-coloring P_6-free graphs. II. Finding an excellent precoloring. *arXiv:1802.02283* (2018).

[8] Díaz, J., Serna, M., and Thilikos, D. M. The restrictive H-coloring problem. *Discrete Applied Mathematics 145* (2005), 297–305.

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

[10] Goldreich, O. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, New York, NY, USA, 2008.

[11] Gutin, G., Hell, P., Rafiey, A., and Yeo, A. A dichotomy for minimum cost graph homomorphisms. *European Journal of Combinatorics 29* (2008), 900–911.

[12] Hell, P., and Nešetřil, J. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B 48* (1990), 92–110.

[13] Hoàng, C. T., Kamiński, M., Lozin, V., Sawada, J., and Shu, X. Deciding k-colorability of P_5-free graphs in polynomial time. *Algorithmica 57* (2010), 74–81.

[14] Holyer, I. The NP-completeness of edge coloring. *SIAM Journal on Computing 10* (1981), 718–720.

[15] Huang, S. Improved complexity results on k-coloring P_t-free graphs. *European Journal of Combinatorics 51* (2016), 336–346.

[16] Impagliazzo, R., and Paturi, R. On the complexity of k-SAT. *Journal of Computer and System Sciences 62* (2001), 367 – 375.

[17] Kamiński, M., and Lozin, V. Coloring edges and vertices of graphs without short or long cycles. *Contributions to Discrete Mathematics 2* (2007), 61–66.

[18] Král, D., Kratochvíl, J., Tuza, Z., and Woeginger, G. J. Complexity of coloring graphs without forbidden induced subgraphs. In *27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)* (2001), A. Brandstädt and V. B. Le, Eds., Springer Berlin Heidelberg, pp. 254–262.

[19] Leven, D., and Galil, Z. NP-completeness of finding the chromatic index of regular graphs. *Journal of Algorithms 4* (1983), 35–44.

[20] Nešetřil, J., and de Mendez, P. Bounded height trees and tree-depth. In *Sparsity: Graphs, Structures, and Algorithms*, vol. 28. Springer, Berlin, Heidelberg, 2012, pp. 115–144.

[21] Scott, A. D., and Sorkin, G. B. Polynomial constraint satisfaction problems, graph bisection, and the Ising partition function. *ACM Trans. Algorithms 5* (2009), 45:1–45:27.

[22] Seymour, P. Tree-chromatic number. *J. Combinatorial Theory, Ser B 116* (2016), 229–237.