Subexponential-Time Algorithms for Finding Large Induced Sparse Subgraphs

Jana Novotná
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
janca@kam.mff.cuni.cz

Karolina Okrasa
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
k.okrasa@mini.pw.edu.pl

Michał Pilipczuk
Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

Paweł Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
p.rzazewski@mini.pw.edu.pl

Erik Jan van Leeuwen
Department of Information and Computing Sciences, Utrecht University, The Netherlands
e.j.vanleeuwen@uu.nl

Bartosz Walczak
Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland
walczak@tcs.uj.edu.pl

Abstract

Let $\mathcal{C}$ and $\mathcal{D}$ be hereditary graph classes. Consider the following problem: given a graph $G \in \mathcal{D}$, find a largest, in terms of the number of vertices, induced subgraph of G that belongs to $\mathcal{C}$. We prove that it can be solved in $2^{o(n)}$ time, where n is the number of vertices of G, if the following conditions are satisfied:

- the graphs in $\mathcal{C}$ are sparse, i.e., they have linearly many edges in terms of the number of vertices;
- the graphs in $\mathcal{D}$ admit balanced separators of size governed by their density, e.g., $\mathcal{O}(\Delta)$ or $\mathcal{O}(\sqrt{m})$, where Δ and m denote the maximum degree and the number of edges, respectively; and
- the considered problem admits a single-exponential fixed-parameter algorithm when parameterized by the treewidth of the input graph.

This leads, for example, to the following corollaries for specific classes $\mathcal{C}$ and $\mathcal{D}$:

- a largest induced forest in a P_t-free graph can be found in $2^{\tilde{\mathcal{O}}(n^{2/3})}$ time, for every fixed t; and
- a largest induced planar graph in a string graph can be found in $2^{\tilde{\mathcal{O}}(n^{3/4})}$ time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Problems, reductions and completeness

Keywords and phrases subexponential algorithm, feedback vertex set, P_t-free graphs, string graphs

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.23

Funding *Jana Novotná*: Supported by student grants GAUK 1277018, SVV-2017-260452.
Michał Pilipczuk: This work is a part of project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).
Paweł Rzążewski: Partially supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.
Bartosz Walczak: Partially supported by Polish National Science Centre grant no. 2015/17/B/ST6/01873.

Acknowledgements The results presented in this paper were obtained during the Parameterized Algorithms Retreat of the algorithms group of the University of Warsaw (PARUW), held in Karpacz in February 2019. This Retreat was financed by the project CUTACOMBS, which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 714704).

1 Introduction

Many optimization problems in graphs can be expressed as follows: given a graph G, find a largest vertex set A such that $G[A]$, the subgraph of G induced by A, satisfies some property. Examples include Independent Set (the property of being edgeless), Feedback Vertex Set (the property of being acyclic), and Planarization (the property of being planar). Here, Feedback Vertex Set and Planarization are customarily phrased in the complementary form that asks for minimizing the complement of A: given G, find a smallest vertex set X such that $G - X$ has the desired property. While all problems considered in this paper can be viewed in these two ways, for the sake of clarity we focus on the maximization formulation.

Formally, we shall consider the following Max Induced $\mathcal{C}$-Subgraph problem. Fix a graph class $\mathcal{C}$ that is *hereditary*, that is, closed under taking induced subgraphs. Then, given a graph G, the goal is to find a largest vertex subset A such that $G[A] \in \mathcal{C}$. Our focus is on exact algorithms for this problem with running time expressed in terms of n, the number of vertices of G. Clearly, as long as the graphs from $\mathcal{C}$ can be recognized in polynomial time, the problem can be solved in $2^n \cdot n^{\mathcal{O}(1)}$ time by brute-force; we are interested in non-trivial improvements over this approach.

The complexity of Max Induced $\mathcal{C}$-Subgraph was studied as early as in 1980 by Lewis and Yannakakis [19], who proved that when the graph class $\mathcal{C}$ does not contain all graphs, the problem is NP-hard. Recently, Komusiewicz [17] inspected the reduction of Lewis and Yannakakis and concluded that under the Exponential Time Hypothesis (ETH) one can even exclude the existence of *subexponential-time* algorithms for the problem, that is, ones with running time $2^{o(n)}$. While the result of Komusiewicz [17] excludes significant improvements in the running time, there is still room for improvement in the base of the exponent. Indeed, for various classes of graphs $\mathcal{C}$, algorithms with running time $\mathcal{O}((2-\varepsilon)^n)$ for some $\varepsilon > 0$ are known; see e.g. [3, 12, 13, 14, 21] and the references therein.

Another direction, which is of main interest to us, is to impose more conditions on the input graphs G in the hope of obtaining faster algorithms for restricted cases. Formally, we fix another hereditary graph class $\mathcal{D}$ and consider Max Induced $\mathcal{C}$-Subgraph where the input graph G is additionally required to belong to $\mathcal{D}$.

In this line of research, the class $\mathcal{C}$ of edgeless graphs, which corresponds to the classical Max Independent Set (MIS) problem, has been extensively studied. Suppose $\mathcal{D}$ is the class of *H-free graphs*, that is, graphs that exclude some fixed graph H as an induced subgraph. As observed by Alekseev [1], the problem is NP-hard on H-free graphs unless H is a path or a subdivision of the claw ($K_{1,3}$); the reduction of [1] actually excludes the existence of a subexponential-time algorithm under ETH in these cases. On the positive side, the maximal classes for which polynomial-time algorithms are known are the P_6-free graphs [15] and the fork-free graphs [20]. It would be consistent with our knowledge if MIS was polynomial-time solvable on H-free graphs whenever H is a path or a subdivision of the claw.

It turns out that if we only aim at subexponential-time instead of polynomial-time algorithms, many more tractability results can be obtained for MIS, and usually they are also much simpler conceptually. Bacsó et al. [2] showed that MIS can be solved in $2^{\mathcal{O}(\sqrt{tn\log n})}$ time on P_t-free graphs, for every $t \in \mathbb{N}$. Very recently, Chudnovsky et al. [8] reported a $2^{\mathcal{O}(\sqrt{n\log n})}$-time algorithm on *long-hole-free graphs*, which are graphs that exclude every cycle of length at least 5 as an induced subgraph.

In the light of the results above, it is natural to ask whether structural assumptions on the class $\mathcal{D}$ from which the input is drawn, like e.g. P_t-freeness, can help in the design of subexponential-time algorithms for other maximum induced subgraph problems, beyond $\mathcal{C}$ being the class of edgeless graphs. This is precisely the question we investigate in this work.

Our contribution. We identify three properties that together provide a way to solve the Max Induced $\mathcal{C}$-Subgraph problem on graphs from $\mathcal{D}$ in subexponential time, where $\mathcal{C}$ and $\mathcal{D}$ are hereditary graph classes. They are as follows:

- The class $\mathcal{C}$ should consist of *sparse* graphs. To be specific, let us assume that every n-vertex graph from $\mathcal{C}$ has $\mathcal{O}(n)$ edges.
- The class $\mathcal{D}$ may contain dense graphs, but they should admit balanced separators whose size is somehow governed by the density. To be specific, let us assume that every graph from $\mathcal{D}$ with maximum degree Δ has a balanced separator of size $\mathcal{O}(\Delta)$, or that every graph from $\mathcal{D}$ with m edges has a balanced separator of size $\mathcal{O}(\sqrt{m})$.
- The Max Induced $\mathcal{C}$-Subgraph problem on graphs from $\mathcal{D}$ can be solved in $2^{\tilde{\mathcal{O}}(w)} \cdot n^{\mathcal{O}(1)}$ time, where w is the treewidth of the input graph. Here, notation $\tilde{\mathcal{O}}(\cdot)$ hides polylogarithmic factors.

We show that if these conditions are simultaneously satisfied, then the Max Induced $\mathcal{C}$-Subgraph problem on graphs from $\mathcal{D}$ can be solved in $2^{\tilde{\mathcal{O}}(n^{2/3})}$ time in the presence of balanced separators of size $\mathcal{O}(\Delta)$ and in $2^{\tilde{\mathcal{O}}(n^{3/4})}$ time for balanced separators of size $\mathcal{O}(\sqrt{m})$. The precise statement and proof of this result can be found in Section 2.

The conditions on $\mathcal{C}$ look natural and are satisfied by various specific classes of interest, like forests (corresponding to Feedback Vertex Set) and planar graphs (corresponding to Planarization). On the other hand, the condition on $\mathcal{D}$ looks more puzzling. However, there are certain non-sparse classes of graphs where the existence of such balanced separators has been established. For instance, balanced separators of size $\mathcal{O}(\Delta)$ are known to exist in P_t-free graphs for any fixed $t \in \mathbb{N}$ [2], and in long-hole-free graphs [8]. The existence of balanced separators of size $\mathcal{O}(\sqrt{m})$ is known for *string graphs*, which are intersection graphs of arc-connected subsets of the plane, and more generally for intersection graphs of connected subgraphs in any proper minor-closed class [18]. All these observations yield a number of concrete corollaries to our main result, which are gathered in Section 3. In Section 4, we discuss some lower bounds: we show that if $\mathcal{C}$ is the class of forests (corresponding to the Feedback Vertex Set problem) and $\mathcal{D}$ is characterized by a single excluded induced subgraph, then under the Exponential Time Hypothesis one cannot hope for subexponential-time algorithms in greater generality than provided by our main result.

2 Main result

We use standard graph notation. We assume the reader's familiarity with treewidth. We recall some notation for tree decompositions in Section 5, where it is actually needed.

For a graph G, a set $S \subseteq V(G)$ is a *balanced separator* if every connected component of $G - S$ has at most $\frac{2}{3}|V(G)|$ vertices. It is known that small balanced separators can be used to construct tree decompositions of small width, as made explicit in the following lemma.

▶ **Lemma 1** ([11]). *If every subgraph of a graph G has a balanced separator of size at most k, then the treewidth of G is $\mathcal{O}(k)$.*

Now, we are ready to state and prove our main result.

▶ **Theorem 2.** *Let $\mathcal{C}$ and $\mathcal{D}$ be classes of graphs that satisfy the following conditions:*

(P1) *Every n-vertex graph from $\mathcal{C}$ has $\mathcal{O}(n)$ edges.*

(P2) *The class $\mathcal{D}$ is closed under taking induced subgraphs.*

(P3) *Given a graph $G \in \mathcal{D}$ with n vertices and treewidth w, one can find a largest set $A \subseteq V(G)$ such that $G[A] \in \mathcal{C}$ in $2^{\tilde{\mathcal{O}}(w)} \cdot n^{\mathcal{O}(1)}$ time.*

Furthermore, let the class $\mathcal{D}$ satisfy one of the following conditions:

(P4a) *Every graph in $\mathcal{D}$ with maximum degree Δ has a balanced separator of size $\mathcal{O}(\Delta)$, or*

(P4b) *Every graph in $\mathcal{D}$ with n vertices and maximum degree Δ has a balanced separator of size $\mathcal{O}(\sqrt{n\Delta})$.*

Then, given an n-vertex graph $G \in \mathcal{D}$, one can find a largest set $A \subseteq V(G)$ such that $G[A] \in \mathcal{C}$ in time

(1) $2^{\tilde{\mathcal{O}}(n^{2/3})}$*, if $\mathcal{D}$ satisfies* (P4a)*, or*

(2) $2^{\tilde{\mathcal{O}}(n^{3/4})}$*, if $\mathcal{D}$ satisfies* (P4b)*.*

Proof. Let a constant τ be defined as follows, depending on which of the two conditions is satisfied by $\mathcal{D}$:

$$\tau = \begin{cases} 1/3 & \text{if } \mathcal{D} \text{ satisfies (P4a)}, \\ 1/4 & \text{if } \mathcal{D} \text{ satisfies (P4b)}. \end{cases}$$

We devise a branching algorithm that finds a largest set $A \subseteq V(G)$ such that $G[A] \in \mathcal{C}$ in $2^{\tilde{\mathcal{O}}(n^{1-\tau})}$ time. This matches the complexity bounds from the statement of the theorem.

Let $G \in \mathcal{D}$ be the input graph and n be the number of its vertices. Consider a fixed solution A, that is, a largest set $A \subseteq V(G)$ such that $G[A] \in \mathcal{C}$. Let $A' \subseteq A$ be the set of vertices of degree greater than n^τ in $G[A]$. By property (P1), we have $|A'| = \mathcal{O}(n/n^\tau) = \mathcal{O}(n^{1-\tau})$.

The algorithm guesses the set A' exhaustively, by trying all subsets of $V(G)$ of the appropriate sizes $\mathcal{O}(n^{1-\tau})$, which results in $n^{\mathcal{O}(n^{1-\tau})} = 2^{\tilde{\mathcal{O}}(n^{1-\tau})}$ branches. Fix one such branch and assume, for the purpose of further description of the algorithm, that it corresponds to the true set A' (i.e., the one obtained from the fixed solution A). Let $G' = G - A'$.

Suppose that G' contains a vertex v of degree at least $n^{2\tau}$. If $v \in A$, then v has degree at most n^τ in $G[A]$ (since $v \notin A'$). The algorithm further guesses that $v \notin A$ and discards v (one branch), or it guesses that $v \in A$ and discards all but at most n^τ neighbors of v in G' (at most n^{n^τ} branches). In the latter case, we do *not* fix the assumption that v or any particular neighbor of v belongs to A, so that the vertices that have survived this step can still be discarded in subsequent branching steps.

The step described above is repeated exhaustively. The overall number of branches generated in this way can be bounded as follows, where $k = |V(G')|$:

$$\begin{aligned} F(k) &\leqslant F(k-1) + n^{n^\tau} \cdot F(k - (n^{2\tau} - n^\tau)) \\ &\leqslant F(k-2) + n^{n^\tau} \cdot F(k - (n^{2\tau} - n^\tau)) + n^{n^\tau} \cdot F(k - (n^{2\tau} - n^\tau)) \\ &\leqslant \ldots \leqslant F(k - (n^{2\tau} - n^\tau)) + (n^{2\tau} - n^\tau) \cdot n^{n^\tau} \cdot F(k - (n^{2\tau} - n^\tau)) \\ &= (n^{2\tau} - n^\tau + 1) \cdot n^{n^\tau} \cdot F(k - (n^{2\tau} - n^\tau)) \\ &\leqslant \left((n^{2\tau} - n^\tau + 1) \cdot n^{n^\tau}\right)^{k/(n^{2\tau} - n^\tau)} \\ &\leqslant \left((n^{2\tau} - n^\tau + 1) \cdot n^{n^\tau}\right)^{n/(n^{2\tau} - n^\tau)} = n^{\mathcal{O}(n^{1+\tau-2\tau})} = 2^{\tilde{\mathcal{O}}(n^{1-\tau})}. \end{aligned}$$

Once the branching step can no longer be applied, we obtain an induced subgraph G'' of G' of maximum degree less than $n^{2\tau}$. In the branch where all the choices have been made correctly (i.e., according to the fixed solution A), G'' still contains all vertices from $A \setminus A'$.

By property (P2), we have $G'' \in \mathcal{D}$. Thus G'' satisfies either (P4a) or (P4b), which means that G'' has a balanced separator of size $\mathcal{O}(n^{2/3})$ in the former case or $\mathcal{O}(\sqrt{n \cdot n^{1/2}}) = \mathcal{O}(n^{3/4})$ in the latter case. In both cases, the size of the separator is $\mathcal{O}(n^{1-\tau})$. Moreover, by the same argument, balanced separators of that size also exist in every subgraph of G''. Therefore, by Lemma 1, we conclude that G'' has treewidth $\mathcal{O}(n^{1-\tau})$. Since $|A'| \leqslant \mathcal{O}(n^{1-\tau})$, it follows that the graph $G[V(G'') \cup A']$ also has treewidth $\mathcal{O}(n^{1-\tau})$.

We know that $G[V(G'') \cup A'] \in \mathcal{D}$ and, in the branch where all choices have been made correctly, this graph contains the entire maximum-size solution A. Now, we apply the procedure assumed in (P3) to the graph $G[V(G'') \cup A']$ and observe that in the correct branch it finds some maximum-size solution (possibly different from A). Let us point out that in this step it is not sufficient to consider only the graph G'', as the vertices from A' introduce some additional constraints on the solution we are looking for.

For the time complexity, the algorithm considers $2^{\tilde{\mathcal{O}}(n^{1-\tau})}$ branches and in each of them it executes the procedure assumed in (P3) in $2^{\tilde{\mathcal{O}}(n^{1-\tau})}$ time, which gives the total running time of $2^{\tilde{\mathcal{O}}(n^{1-\tau})}$. ◀

▶ Remark 3. The condition (P1) in the statement of Theorem 2 can be relaxed to "every n-vertex graph from $\mathcal{C}$ has $\mathcal{O}(n^{2-\varepsilon})$ edges, for some constant $\varepsilon > 0$". Then, we can follow the same approach with the following modification: we choose $\tau = 1 - \frac{2}{3}\varepsilon$ in case of (P4a) and $\tau = 1 - \frac{3}{4}\varepsilon$ in case of (P4b), and replace the threshold for branching on high-degree vertices from $n^{2\tau}$ to $n^{2\tau+\varepsilon-1}$. This way, we obtain algorithms with running time $2^{\tilde{\mathcal{O}}(n^{1-\varepsilon/3})}$ for property (P4a) and $2^{\tilde{\mathcal{O}}(n^{1-\varepsilon/4})}$ for property (P4b). This running time is subexponential for every $\varepsilon > 0$.

One can also imagine unifying properties (P4a) and (P4b) into the existence of a balanced separator of size $\mathcal{O}(n^{\alpha}\Delta^{\beta})$, for some constants α, β. However, then, one needs to be careful when choosing τ so that it belongs to the interval $[0, 1]$. As we did not find concrete examples of interesting graph classes $\mathcal{D}$ for which this approach would yield non-trivial results and which would not satisfy either (P4a) or (P4b), we refrain from discussing further details here.

3 Corollaries

In this section, we discuss possible classes $\mathcal{C}$ and $\mathcal{D}$ which satisfy the conditions of Theorem 2. For some choices of $\mathcal{C}$, we obtain well-studied computational problems:

1. for matchings, we obtain Max Induced Matching,
2. for forests, we obtain Max Induced Forest, also known as Feedback Vertex Set,
3. for graphs of maximum degree d, where d is fixed, we obtain Max Induced Degree-d Subgraph,
4. for planar graphs, we obtain Max Induced Planar Subgraph, also known as Planarization,
5. for graphs embeddable in Σ, where the surface Σ is fixed, we obtain Max Induced Σ-Embeddable Subgraph,
6. for graphs of degeneracy at most d, where d is fixed, we obtain Max Induced d-Degenerate Subgraph.

It is clear that all these classes satisfy property 1 of Theorem 2.

Given a graph of treewidth w, its tree decomposition of width at most $4w + 3$ can be computed in $2^{\mathcal{O}(w)} \cdot n^2$ time (see e.g. [9, Section 7.6]). Therefore, for the purpose of verifying property 3, we can assume that a tree decomposition of width $\mathcal{O}(w)$ is additionally provided on input. While $2^{\tilde{\mathcal{O}}(w)} \cdot n^{\mathcal{O}(1)}$-time algorithms are quite straightforward and well known for the first two problems on the list, this is not necessarily the case for the others. For Max Induced Degree-d Subgraph, an algorithm with running time $2^{\mathcal{O}(w)} \cdot n^{\mathcal{O}(1)}$ can be easily derived from the meta-theorem of Pilipczuk [22]. Algorithms for Max Induced Planar Subgraph and, more generally, Max Induced Σ-Embeddable Subgraph, were provided by Kociumaka and Pilipczuk [16]. Finally, we give a suitable algorithm for Max Induced d-Degenerate Subgraph in Lemma 8 in Section 5.

It may be tempting to consider, as $\mathcal{C}$, the graphs with no even cycle C_{2k} (not necessarily induced), for some fixed integer $k \geqslant 2$. This is because such graphs have $\mathcal{O}(n^{2-\Omega(1/k)})$ edges [5], and thus they satisfy the generalization of property 1 mentioned in Remark 3 for $\varepsilon = \Omega(1/k)$. However, for these classes, property 3 turns out to be problematic: for any fixed $\ell \geqslant 5$, there is no algorithm for a minimum set of vertices hitting all (non-induced) copies of C_ℓ in a graph with treewidth w with running time $2^{o(w^2)} \cdot n^{\mathcal{O}(1)}$ unless the ETH fails [22] (this bound appears to be essentially tight, as the problem can be solved in $2^{\tilde{\mathcal{O}}(w^2)} \cdot n^{\mathcal{O}(1)}$ time [10]). It is unclear whether the additional assumption that the input graph belongs to some class $\mathcal{D}$, considered here, can help.

Now, let us consider classes $\mathcal{D}$. Examples of classes satisfying property a in Theorem 2 come from forbidding some induced subgraphs. Bacsó et al. [2] proved that P_t-free graphs with maximum degree Δ have treewidth $\mathcal{O}(\Delta \cdot t)$. Very recently, Chudnovsky et al. [8] observed that *long-hole-free graphs*, that is, graphs with no induced cycles of length at least 5, also have balanced separators of size $\mathcal{O}(\Delta)$.

An example of a class satisfying property b is the class of string graphs – intersection graphs of arc-connected subsets of the plane. Lee [18] showed that they admit balanced separators of size $\mathcal{O}(\sqrt{m})$, where m is the number of edges. In fact, he proved a more general result that if $\mathcal{M}$ is a class of graphs excluding a fixed graph as a minor, then intersection graphs of connected subgraphs of graphs from $\mathcal{M}$ admit balanced separators of size $\mathcal{O}(\sqrt{m})$. String graphs are precisely the intersection graphs of connected subgraphs of planar graphs.

Summing up, we obtain the following.

▶ **Corollary 4.** *Each of the following problems can be solved in $2^{\tilde{\mathcal{O}}(n^{2/3})}$ time on P_t-free graphs (for every fixed t) and in long-hole-free graphs, and in $2^{\tilde{\mathcal{O}}(n^{3/4})}$ time on string graphs:*
1. Max Induced Matching,
2. Max Induced Forest,
3. Max Induced Degree-d Subgraph, *for every fixed $d \in \mathbb{N}$,*
4. Max Induced Planar Subgraph,
5. Max Induced Σ-Embeddable Subgraph, *for every fixed surface Σ,*
6. Max Induced d-Degenerate Subgraph, *for every fixed $d \in \mathbb{N}$.*

We note that subexponential-time algorithms for Max Induced Matching and Max Induced Forest on string graphs were already known [6], even with a better running time than provided above. As we have argued, in Corollary 4, we can replace string graphs with intersection graphs of connected subgraphs of graphs from $\mathcal{M}$, where $\mathcal{M}$ is any class of graphs excluding a fixed graph as a minor; this is because the result of Lee [18] holds in that generality.

4 Max Induced Forest in H-free graphs

Our original motivation was the Max Induced Forest problem. In the previous section, we discussed a subexponential-time algorithm solving it on P_t-free graphs. We now show that as long as the considered class of inputs $\mathcal{D}$ is characterized by a single excluded induced subgraph, that is, we investigate Max Induced Forest on H-free graphs for a fixed graph H, we cannot hope for more positive results. Namely, it turns out that if H is not a linear forest (i.e., a collection of vertex-disjoint paths), the problem is unlikely to admit a polynomial-time or even a subexponential-time algorithm on H-free graphs. Specifically, we obtain the following dichotomy.

▶ **Theorem 5.** *Let H be a fixed graph.*

1. *If H is a linear forest, then the* Max Induced Forest *problem can be solved in $2^{\tilde{O}(n^{2/3})}$ time on H-free graphs with n vertices.*
2. *Otherwise, on H-free graphs, the* Max Induced Forest *problem is NP-complete and cannot be solved in $2^{o(n)}$ time unless the ETH fails.*

Theorem 5 1 follows from Corollary 4, because every linear forest is an induced subgraph of some path. Statement 2 follows from a combination of arguments already existing in the literature. However, since the proof is simple, we include it for the sake of completeness.

We prove Theorem 5 2 in two steps. First, we consider graphs H that contain a cycle or two branch vertices, that is, vertices of degree at least 3. In this case, we can apply the standard argument of subdividing every edge a suitable number of times, cf. [7, Theorem 3].

▶ **Lemma 6.** *Let H be a fixed graph that either contains a cycle or has a connected component with at least two branch vertices. Then* Max Induced Forest *is NP-complete on H-free graphs. Moreover, there is no algorithm solving* Max Induced Forest *in $2^{o(n)}$ time for n-vertex H-free graphs unless the ETH fails.*

Proof. We reduce from Max Induced Forest in graphs with maximum degree 6; it is known that this problem is NP-complete and has no subexponential-time algorithm assuming ETH [9]. Let G be a graph with n vertices and maximum degree 6. Let G^* be the graph obtained from G by subdividing every edge $|V(H)|+1$ times. It is straightforward to observe that G has an induced forest on $n-k$ vertices if and only if G^* has an induced forest on $|G^*|-k$ vertices. Moreover, the number of vertices in G^* is linear in n.

Finally, we show that G^* is H-free. First, observe that if H contains a cycle, then H cannot be a subgraph of G^*, as the girth of G^* is greater than $|V(H)|+1$. On the other hand, the distance between any two branch vertices in G^* is at least $|V(H)|+1$, so G^* does not contain H as a subgraph in case H has two branch vertices in the same connected component. ◀

By Lemma 6, the only graphs H for which we might hope for a polynomial-time or even a subexponential-time algorithm for Max Induced Forest on H-free graphs are collections of disjoint subdivided stars. To resolve this case, we will show that the problem remains hard for line graphs. Recall that the line graph $L(G)$ of a graph G is the graph whose vertices are the edges of G and where the adjacency relation corresponds to the relation of having a common endpoint in G.

Actually, Chiarelli et al. [7] reported that the hardness of Max Induced Forest on line graphs was observed by Speckenmeyer in his PhD thesis [23]. However, we were unable to find this result there. Therefore, we provide the easy proof, which boils down to essentially the same argument as in [7, Theorem 5].

▶ **Lemma 7.** Max Induced Forest *is NP-complete on line graphs. Moreover, there is no algorithm solving* Max Induced Forest *in $2^{o(n)}$ time for n-vertex line graphs unless the ETH fails.*

Proof. We reduce from the Hamiltonian Path problem, which is NP-complete and has no subexponential-time algorithm, even if the input graph has linearly many edges [9]. Let G be a graph, which is the input instance of Hamiltonian Path.

First, note that any induced forest in $L(G)$ corresponds to a collection of vertex-disjoint paths in G. More formally, consider a set $E' \subseteq E(G)$, such that $L(G)[E']$ is a forest. We claim that the subgraph $G' = (V(G), E')$ of G is a collection of vertex-disjoint paths. Suppose not. This means that G' contains a vertex v of degree at least 3 or a cycle C. In the former case, the edges incident to v in G' form a clique in $L(G)[E']$. In the latter case, the edges of the cycle C form a cycle in $L(G)[E']$. In either case, we get a contradiction to the assumption that $L(G)[E']$ is a forest.

We claim that G has a Hamiltonian path if and only if $L(G)$ has an induced forest on $n-1$ vertices. Indeed, the $n-1$ edges of a Hamiltonian path in G induce a path (in particular, a forest) in $L(G)$. For the converse, suppose that $L(G)$ has an induced forest on at least $n-1$ vertices. By the observation above, this induced forest corresponds to a collection of vertex-disjoint paths in G with at least $n-1$ edges in total. This is only possible if this collection consists of a single path of length $n-1$, that is, a Hamiltonian path in G.

Finally, observe that the number of vertices of $L(G)$ is equal to the number of edges of G, which is linear in the number of vertices of G. ◀

Recall that line graphs are claw-free, that is, they contain no induced copy of $K_{1,3}$. Thus Lemma 7 implies that if H contains any star with at least 3 leaves, then Max Induced Forest remains NP-complete and has no subexponential-time algorithm on H-free graphs unless ETH fails. Theorem 5 2 follows from combining Lemma 6 and Lemma 7.

5 Largest induced degenerate subgraph in low-treewidth graphs

This section is devoted to the proof of the following result, which we used in Section 3.

▶ **Lemma 8.** *For every fixed $d \in \mathbb{N}$, there is an algorithm for* Max Induced d-Degenerate Subgraph *with running time $2^{\mathcal{O}(w \log w)} \cdot n$, where w is the treewidth of the input graph and n is the number of its vertices.*

Preliminaries on tree decompositions. First, we introduce some notation and terminology. A *tree decomposition* of a graph G is a tree T together with a mapping $\beta(\cdot)$ that assigns a *bag* $\beta(x)$ to each node x of T in such a way that the following conditions hold:

(T1) for each $u \in V(G)$, the set of nodes x with $u \in \beta(x)$ induces a connected non-empty subtree of T; and

(T2) for each $uv \in E(G)$, there exists a node x such that $\{u, v\} \subseteq \beta(x)$.

The *width* of a tree decomposition (T, β) is $\max_{x \in V(T)} |\beta(x)| - 1$, and the *treewidth* of a graph G is the minimum width of a tree decomposition of G.

Henceforth, all tree decompositions will be *rooted*: the underlying tree T has a prescribed root vertex r. This gives rise a natural ancestor-descendant relation: we write $x \preceq y$ if x is an ancestor of y (where possibly $x = y$). Then, for a node x of T, we define the *component* at x as

$$\alpha(x) = \left(\bigcup_{y \succeq x} \beta(y) \right) \setminus \beta(x).$$

It easily follows from (T1) and (T2) that then $N(\alpha(x)) \subseteq \beta(x)$ for every node x.

A *nice tree decomposition* is a normalized form of a rooted tree decomposition in which every node is of one of the following four kinds.

- **Leaf node**: a node x with no children and with $\beta(x) = \emptyset$.
- **Introduce node**: a node x with one child y such that $\beta(x) = \beta(y) \cup \{u\}$ for some vertex $u \notin \beta(y)$.
- **Forget node**: a node x with one child y such that $\beta(x) = \beta(y) \setminus \{u\}$ for some vertex $u \in \beta(y)$.
- **Join node**: a node x with two children y and z such that $\beta(x) = \beta(y) = \beta(z)$.

Moreover, we require that the root r of the nice tree decomposition satisfies $\beta(r) = \emptyset$.

It is known that any given tree decomposition (T, β) of width k of an n-vertex graph G can be transformed in $k^{\mathcal{O}(1)} \cdot \max(n, |V(T)|)$ time into a nice tree decomposition of G of width at most as large, see [9, Lemma 7.4]. Moreover, given an n-vertex graph G of treewidth w, a tree decomposition of G of width at most $5w+4$ can be computed in $2^{\mathcal{O}(w)} \cdot n$ time [4], and this tree decomposition has at most n nodes. By combining these two results, for the proof of Lemma 8, we can assume that the input graph G is supplied with a nice tree decomposition (T, β) of width $k \leqslant 5w + 4$, where $w = \mathrm{tw}(G)$. From now on, our goal is to design a suitable dynamic programming algorithm working on this decomposition with running time $2^{\mathcal{O}(k \log k)} \cdot n = 2^{\mathcal{O}(w \log w)} \cdot n$.

Dynamic programming states. The main idea behind our dynamic programming algorithm is to view the notion of degeneracy via vertex orderings, as expressed in the following fact.

▶ **Lemma 9** (Folklore). *A graph H is d-degenerate if and only if there is a linear ordering σ of vertices of H such that every vertex of H has at most d neighbors that are smaller in σ.*

Hence, the problem considered in Lemma 8 can be restated as follows: find a largest set $A \subseteq V(G)$ that admits a linear ordering σ in which every vertex of A has at most d neighbors in $G[A]$ that are smaller in σ. Intuitively, our dynamic programming will therefore keep track of the intersection of the bag with A, the restriction of σ to this intersection; and how many smaller neighbors of each vertex from this intersection have been already forgotten.

We now proceed with formal details. For a node x of T, a set $X \subseteq \beta(x)$, a linear ordering σ of X, and a function $f \colon X \to \{0, \ldots, d\}$, we define $\Phi_x[X, \sigma, f] \in \mathbb{N}$ as follows. The value $\Phi_x[X, \sigma, f]$ is the maximum size of a set $Y \subseteq \alpha(x)$ such that $X \cup Y$ admits a linear ordering τ with the following properties: τ restricted to X is equal to σ and for every $a \in X$, there are at most $f(a)$ vertices $b \in Y$ that are adjacent to a and smaller than a in τ. Note that other neighbors of a that belong to X are *not* taken into consideration when verifying the quota imposed by $f(a)$. Note also that such a set Y always exists, as $Y = \emptyset$ satisfies the criteria.

For a fixed node x, the total number of triples (X, σ, f) as above is at most

$$2^{k+1} \cdot (k+1)! \cdot (d+1)^{k+1} \leqslant 2^{\mathcal{O}(k \log k)}.$$

Hence, we now show how to compute the values $\Phi_x[X, \sigma, f]$ in a bottom-up manner, so that the values for a node x are computed based on the values for the children of x in $2^{\mathcal{O}(k \log k)}$ time. The answer to the problem corresponds to the value $\Phi_r[\emptyset, \emptyset, \emptyset]$, where r is the root of T. While $\Phi_r[\emptyset, \emptyset, \emptyset]$ is just the size of a largest feasible solution, an actual solution can be recovered from the dynamic programming tables using standard methods within the same complexity: for every computed value $\Phi_x[X, \sigma, f]$, we store the way this value was obtained, and then we trace back the solution from $\Phi_r[\emptyset, \emptyset, \emptyset]$ in a top-down manner.

Transitions. It remains to provide recursive formulas for the values of $\Phi_x[\cdot, \cdot, \cdot]$. We only present the formulas, while the verification of their correctness, which follows easily from the definition of $\Phi_x[\cdot, \cdot, \cdot]$, is left to the reader. As usual, we distinguish cases depending on the type of x.

- **Leaf node** x. Then we have only one value:

  $$\Phi_x[\emptyset, \emptyset, \emptyset] = 0.$$

- **Introduce node** x with child y such that $\beta(x) = \beta(y) \cup \{u\}$. Then

  $$\Phi_x[X, \sigma, f] = \begin{cases} \Phi_y[X, \sigma, f] & \text{if } u \notin X; \\ \Phi_y[X \setminus \{u\}, \sigma|_{X\setminus\{u\}}, f|_{X\setminus\{u\}}] & \text{if } u \in X. \end{cases}$$

- **Forget node** x with child y such that $\beta(x) = \beta(y) \setminus \{u\}$. Then we have

  $$\Phi_x[X, \sigma, f] = \max \left(\Phi_y[X, \sigma, f],\ 1 + \max_{(\sigma', f') \in S(X, \sigma, f)} \Phi_y[X \cup \{u\}, \sigma', f'] \right),$$

  where $S(X, \sigma, f)$ is the set comprising the pairs (σ', f') satisfying the following:
  - σ' is a vertex ordering of $X \cup \{u\}$ whose restriction to X is equal to σ; and
  - $f' \colon X \cup \{u\} \to \{0, \dots, d\}$ is such that for all $a \in X$ that are adjacent to u and larger than u in σ', we have $f'(a) \leqslant f(a) - 1$, and for all other $a \in X$, we have $f'(a) \leqslant f(a)$. Moreover, we require that $f'(u) \leqslant d - \ell$, where ℓ is the number of vertices $a \in X$ that are adjacent to u and smaller than u in σ'.
- **Join node** x with children y and z. Then

  $$\Phi_x[X, \sigma, f] = \max_{f_y + f_z \leqslant f} \Phi_y[X, \sigma, f_y] + \Phi_z[X, \sigma, f_z],$$

  where $f_y + f_z \leqslant f$ means that $f_y(a) + f_z(a) \leqslant f(a)$ for each $a \in X$.

It is straightforward to see that using the formulas above, each value $\Phi_x[X, \sigma, f]$ can be computed in $2^{\mathcal{O}(k \log k)}$ time based on the values computed for the children of x. This completes the proof of Lemma 8.

References

1 Vladimir E. Alekseev. The effect of local constraints on the complexity of determination of the graph independence number. *Combinatorial-algebraic methods in applied mathematics*, pages 3–13, 1982. (in Russian).

2 Gábor Bacsó, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Zsolt Tuza, and Erik Jan van Leeuwen. Subexponential-Time Algorithms for Maximum Independent Set in P_t-Free and Broom-Free Graphs. *Algorithmica*, 81(2):421–438, 2019.

3 Ivan Bliznets, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Largest Chordal and Interval Subgraphs Faster than 2^n. *Algorithmica*, 76(2):569–594, 2016. doi:10.1007/s00453-015-0054-2.

4 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

5 John Adrian Bondy and Miklós Simonovits. Cycles of even length in graphs. *J. Combin. Theory Ser. B*, 16(2):97–105, 1974.

6 Édouard Bonnet and Paweł Rzążewski. Optimality Program in Segment and String Graphs. *Algorithmica*, 81(7):3047–3073, 2019. doi:10.1007/s00453-019-00568-7.

7 Nina Chiarelli, Tatiana Romina Hartinger, Matthew Johnson, Martin Milanič, and Daniël Paulusma. Minimum connected transversals in graphs: New hardness results and tractable cases using the price of connectivity. *Theor. Comput. Sci.*, 705:75–83, 2018. doi:10.1016/j.tcs.2017.09.033.

8 Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. On the Maximum Weight Independent Set Problem in graphs without induced cycles of length at least five. *CoRR*, abs/1903.04761, 2019. `arXiv:1903.04761`.

9 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

10 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Inf. Comput.*, 256:62–82, 2017. `doi:10.1016/j.ic.2017.04.009`.

11 Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *J. Combin. Theory Ser. B*, 137:137–144, 2019. `doi:10.1016/j.jctb.2018.12.007`.

12 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. In *STOC 2016*, pages 764–775. ACM, 2016.

13 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Exact Algorithm for the Maximum Induced Planar Subgraph Problem. In *ESA 2011*, volume 6942 of *LNCS*, pages 287–298. Springer, 2011.

14 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large Induced Subgraphs via Triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015.

15 Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. Polynomial-time algorithm for Maximum Weight Independent Set on P_6-free graphs. In *SODA 2019*, pages 1257–1271. SIAM, 2019.

16 Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. *CoRR*, abs/1706.04065, 2017. `arXiv:1706.04065`.

17 Christian Komusiewicz. Tight Running Time Lower Bounds for Vertex Deletion Problems. *ACM Trans. on Comput. Theory (TOCT)*, 10(2):6:1–6:18, 2018.

18 James R. Lee. Separators in Region Intersection Graphs. In *ITCS 2017*, volume 67 of *LIPIcs*, pages 1:1–1:8. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2017.

19 John M. Lewis and Mihalis Yannakakis. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. `doi:10.1016/0022-0000(80)90060-4`.

20 Vadim V. Lozin and Martin Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *J. Discrete Algorithms*, 6(4):595–604, 2008.

21 Marcin Pilipczuk and Michał Pilipczuk. Finding a Maximum Induced Degenerate Subgraph Faster Than 2^n. In *IPEC 2012*, volume 7535 of *LNCS*, pages 3–12. Springer, 2012.

22 Michał Pilipczuk. Problems Parameterized by Treewidth Tractable in Single Exponential Time: A Logical Approach. In *MFCS 2011*, volume 6907, pages 520–531. Springer, 2011.

23 Ewald Speckenmeyer. *Untersuchungen zum Feedback Vertex Set Problem in ungerichteten Graphen*. PhD thesis, Universität Paderborn, 1983. In German.