Finding Large Induced Sparse Subgraphs in $C_{>t}$-Free Graphs in Quasipolynomial Time*

Peter Gartland
petergartland@ucsb.edu
University of California
Santa Barbara, USA

Daniel Lokshtanov
daniello@ucsb.edu
University of California
Santa Barbara, USA

Marcin Pilipczuk
malcin@mimuw.edu.pl
Institute of Informatics,
University of Warsaw
Warsaw, Poland

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

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

ABSTRACT

For an integer t, a graph G is called *$C_{>t}$-free* if G does not contain any induced cycle on more than t vertices. We prove the following statement: for every pair of integers d and t and a CMSO_2 statement φ, there exists an algorithm that, given an n-vertex $C_{>t}$-free graph G with weights on vertices, finds in time $n^{O(\log^3 n)}$ a maximum-weight vertex subset S such that $G[S]$ has degeneracy at most d and satisfies φ. The running time can be improved to $n^{O(\log^2 n)}$ assuming G is P_t-free, that is, G does not contain an induced path on t vertices. This expands the recent results of the authors [FOCS 2020 and SOSA 2021] on the Maximum Weight Independent Set problem on P_t-free graphs in two directions: by encompassing the more general setting of $C_{>t}$-free graphs, and by being applicable to a much wider variety of problems, such as Maximum Weight Induced Forest or Maximum Weight Induced Planar Graph.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Mathematics of computing → Graph theory**.

KEYWORDS

independent set, feedback vertex set, P_t-free graphs, $C_{>t}$-free graphs

ACM Reference Format:
Peter Gartland, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. 2021. Finding Large Induced Sparse Subgraphs in $C_{>t}$-Free Graphs in Quasipolynomial Time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3406325.3451034

*Due to space limits, most of technicals details are omitted or just sketched. The full version of the paper is available on arXiv [14].

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451034

1 INTRODUCTION

Consider the Maximum Weight Independent Set (MWIS) problem: given a vertex-weighted graph G, find an independent set in G that has the largest possible weight. While NP-hard in general, the problem becomes more tractable when structural restrictions are imposed on the input graph G. In this work we consider restricting G to come from a fixed *hereditary* (closed under taking induced subgraphs) class C. The goal is to understand how the complexity of MWIS, and of related problems, changes with the class C. A concrete instance of this question is to consider *H-free graphs* — graphs that exclude a fixed graph H as an induced subgraph — and classify for which H, MWIS becomes polynomial-time solvable in H-free graphs.

Somewhat surprisingly, we still do not know the complete answer to this question. A classic argument of Alekseev [2] shows that MWIS is NP-hard in H-free graphs, unless H is a forest of paths and *subdivided claws*: graphs obtained from the claw $K_{1,3}$ by subdividing each of its edges an arbitrary number of times. The remaining cases are still open apart from several small ones: of P_5-free graphs [19], P_6-free graphs [16], claw-free graphs [21, 26], and fork-free graphs [3, 20]. Here and further on, P_t denotes a path on t vertices.

On the other hand, there are multiple indications that MWIS indeed has a much lower complexity in H-free graphs, whenever H is a forest of paths and subdivided claws, than in general graphs. Concretely, in this setting the problem is known to admit both a subexponential-time algorithm [5, 7] and a QPTAS [7, 8]; note that the existence of such algorithms for general graphs is excluded under standard complexity assumptions. Very recently, the first two authors gave a *quasipolynomial-time* algorithm for MWIS in P_t-free graphs, for every fixed t [13]. The running time was $n^{O(\log^3 n)}$,

which was subsequently improved to $n^{O(\log^2 n)}$ by the last three authors [24].

A key fact that underlies most of the results stated above is that P_t-free graphs admit the following balanced separator theorem (see Theorem 2.1): In every P_t-free graph, we can find a connected set X consisting of at most t vertices, such that the number of vertices in every connected component of $G - N[X]$ is at most half of the number of vertices of G. It has been observed by Chudnovsky et al. [8] that the same statement is true also in the class of *$C_{>t}$-free graphs*: graphs that do not contain an induced cycle on more than t vertices. Note here that, on one hand, every P_t-free graph is $C_{>t}$-free as well, and, on the other hand, $C_{>t}$-free graphs generalize the well-studied class of *chordal graphs*, which are exactly $C_{>3}$-free. Using the separator theorem, Chudnovsky et al. [7, 8] gave a subexponential-time algorithm and a QPTAS for MWIS on $C_{>t}$-free graphs, for every fixed t.

The basic toolbox developed for MWIS can also be applied to other problems of similar nature. Consider, for instance, the Maximum Weight Induced Forest problem: in a given vertex-weighted graph G, find a maximum-weight vertex subset that induces a forest; note that by duality, this problem is equivalent to Feedback Vertex Set. By lifting techniques used to solve MWIS in polynomial time in P_5-free and P_6-free graphs [16, 19], Abrishami et al. [1] showed that Maximum Weight Induced Forest is polynomial-time solvable both in P_5-free and in $C_{>4}$-free graphs. In fact, the result is even more general: it applies to every problem of the form "find a maximum-weight induced subgraph of treewidth at most k"; MWIS and Maximum Weight Induced Forest are particular instantiations for $k = 0$ and $k = 1$, respectively.

As far as subexponential-time algorithms are concerned, Novotná et al. [23] showed how to use separator theorems to get subexponential-time algorithms for any problem of the form "find the largest induced subgraph belonging to $\mathcal{C}$", where $\mathcal{C}$ is a fixed hereditary class of graphs that have a linear number of edges. The technique applies both to P_t-free and $C_{>t}$-free graphs under the condition that the problem in question admits an algorithm which is single-exponential in the treewidth of the instance graph.

Our results. We extend the recent results on quasipolynomial-time algorithms for MWIS in P_t-free graphs [13, 24] in two directions:

(a) We expand the area of applicability of the techniques to $C_{>t}$-free graphs.
(b) We show how to solve in quasipolynomial time not only the MWIS problems, but a whole family of problems that can be, roughly speaking, described as finding a maximum-weight induced subgraph that is sparse and satisfies a prescribed property.

Both of these extensions require a significant number of new ideas. Formally, we prove the following.

Theorem 1.1. *Fix a pair of integers d and t and a CMSO_2 sentence φ. Then there exists an algorithm that, given a $C_{>t}$-free n-vertex graph G and a weight function $\mathfrak{w}\colon V(G) \to \mathbb{N}$, in time $n^{O(\log^3 n)}$ finds a subset S of vertices such that $G[S]$ is d-degenerate, $G[S]$ satisfies φ, and, subject to the above, $\mathfrak{w}(S)$ is maximum possible; the algorithm may also conclude that no such vertex subset exists. The running time can be improved to $n^{O(\log^2 n)}$ if G is P_t-free.*

Recall here that a graph G is *d-degenerate* if every subgraph of G contains a vertex of degree at most d; for instance, 1-degenerate graphs are exactly forests and every planar graph is 5-degenerate. Also, CMSO_2 is the *Monadic Second Order* logic of graphs with quantification over edge subsets and modular predicates, which is a standard logical language for formulating graph properties. In essence, the logic allows quantification over single vertices and edges as well as over subsets of vertices and of edges. In atomic expressions one can check whether an edge is incident to a vertex, whether a vertex/edge belongs to a vertex/edge subset, and whether the cardinality of some set is divisible by a fixed modulus. We refer to [10] for a broader introduction.

Corollaries. By applying Theorem 1.1 for different sentences φ, we can model various problems of interest. For instance, as 1-degenerate graphs are exactly forests, we immediately obtain a quasipolynomial-time algorithm for the Maximum Weight Induced Forest problem in $C_{>t}$-free graphs. Further, as being planar is expressible in CMSO_2 and planar graphs are 5-degenerate, we can conclude that the problem of finding a maximum-weight induced planar subgraph can be solved in quasipolynomial time on $C_{>t}$-free graphs. We also give a generalization of Theorem 1.1 that allows counting the weights only on a subset of S. From this generalization it follows that for instance the following problem can be solved in quasipolynomial time on $C_{>t}$-free graphs: find the largest collection of pairwise nonadjacent induced cycles.

Let us point out a particular corollary of Theorem 1.1 of a more general nature. It is known that for every pair of integers d and t there exists $\ell = \ell(d,t)$ such that every graph that contains P_ℓ *as a subgraph*, contains either K_{d+2}, or $K_{d+1,d+1}$, or P_t as an induced subgraph [4]. Since the degeneracy of K_{d+2} and $K_{d+1,d+1}$ is larger than d, we conclude that every P_t-free graph of degeneracy at most d does not contain P_ℓ as a subgraph. On the other hand, for every integer ℓ, the class of graphs that do not contain P_ℓ as a subgraph is well-quasi-ordered by the induced subgraph relation [11]. It follows that for every pair of integers t and d and every hereditary class $\mathcal{C}_d$ such that every graph in $\mathcal{C}_d$ has degeneracy at most d, the class $\mathcal{C}_d \cap (P_t\text{-free})$ of P_t-free graphs from $\mathcal{C}_d$ is characterized by a finite number of forbidden induced subgraphs: there exists a finite list $\mathcal{F}$ of graphs such that a graph G belongs to $\mathcal{C}_d \cap (P_t\text{-free})$ if and only if G does not contain any graph from $\mathcal{F}$ as an induced subgraph. As admitting a graph from $\mathcal{F}$ as an induced subgraph can be expressed by a CMSO_2 sentence, from Theorem 1.1 we can conclude the following.

Theorem 1.2. *Let $\mathcal{C}$ be a hereditary graph class such that each member of $\mathcal{C}$ is d-degenerate, for some integer d. Then for every integer t there exists algorithm that, given a P_t-free n-vertex graph G and a weight function $\mathfrak{w}\colon V(G) \to \mathbb{N}$, in time $n^{O(\log^2 n)}$ finds a subset S of vertices such that $G[S] \in \mathcal{C}$ and, subject to this, $\mathfrak{w}(S)$ is maximum possible.*

Degeneracy and treewidth. Readers familiar with the literature on algorithmic results for CMSO_2 logic might be slightly surprised by the statement of Theorem 1.1. Namely, CMSO_2 is usually associated with graphs of bounded treewidth, where the tractability of

problems expressible in this logic is asserted by Courcelle's Theorem [9]. Theorem 1.1, however, speaks about CMSO_2-expressible properties of graphs of bounded degeneracy. While degeneracy is upper-bounded by treewidth, in general there are graphs that have bounded degeneracy and arbitrarily high treewidth. However, we prove that in the case of $C_{>t}$-free graphs, the two notions are functionally equivalent.

Theorem 1.3. *For every pair of integers d and t, there exists an integer $k = (dt)^{O(t)}$ such that every $C_{>t}$-free graph of degeneracy at most d has treewidth at most k.*

As the properties of having treewidth at most k and having degeneracy at most d are expressible in CMSO_2, from Theorem 1.3 it follows that in the statement of Theorem 1.1, assumptions "$G[S]$ has degeneracy at most d" and "$G[S]$ has treewidth at most k" could be replaced by one another. Actually, both ways of thinking will become useful in the proof.

Simple QPTASes. As an auxiliary result, we also show a simple technique for turning algorithms for MWIS in P_t-free and $C_{>t}$-free graphs into approximation schemes for (unweighted) problems of the following form: in a given graph, find the largest induced subgraph belonging to C, where C is a fixed graph class that is closed under taking disjoint unions and induced subgraphs and is *weakly hyperfinite* [22, Section 16.2]. This last property is formally defined as follows: for every $\varepsilon > 0$, there exists a constant $c(\varepsilon)$ such that from every graph $G \in C$ one can remove an ε fraction of vertices so that every connected component of the remaining graph has at most $c(\varepsilon)$ vertices. Weak hyperfiniteness is essentially equivalent to admitting sublinear balanced separators, so all the well-known classes of sparse graphs, e.g. planar graphs or all proper minor-closed classes, are weakly hyperfinite. We present these results in Section 4.

3-Coloring. In [24], it is shown how to modify the quasipolynomial-time algorithm for MWIS in P_t-free graphs to obtain an algorithm for 3-Coloring with the same asymptotic running time bound in the same graph class. We remark here that the same modification can be applied to the algorithm of Theorem 1.1, obtaining the following:

Theorem 1.4. *For every integer t there exists an algorithm that, given an n-vertex $C_{>t}$-free graph G, runs in time $n^{O(\log^3 n)}$ and verifies whether G is 3-colorable.*

2 OVERVIEW OF THE MAIN RESULT

In this section we present an overview of the proof of our main result, Theorem 1.1. We try to keep the description non-technical, focusing on explaining the main ideas and intuitions. Complete and formal proofs can be found in the full version of the paper [14].

2.1 Approach for P_t-Free Graphs

We need to start by recalling the basic idea of the quasipolynomial-time algorithm for MWIS in P_t-free graphs [13, 24]; we choose to follow the exposition of [24]. The main idea is to exploit the following balanced separator theorem.

Theorem 2.1 (Gyárfás [17], Bacsó et al. [5]). *Let G be an n-vertex P_t-free graph. Then there exists a set X consisting of at most t vertices of G such that $G[X]$ is connected and every connected component of $G - N[X]$ has at most $n/2$ vertices. Furthermore, such a set can be found in polynomial time.*

In the MWIS problem, there is a natural branching strategy that can be applied on any vertex u. Namely, branch into two subproblems: in one subproblem — *success branch* — assume that u is included in an optimal solution, and in the other — *failure branch* — assume it is not. In the success branch we can remove both u and all its neighbors from the consideration, while in the failure branch only u can be removed. Hence, Theorem 2.1 suggests the following naive Divide&Conquer strategy: find a set X as provided by the Theorem and branch on all the vertices of X as above in order to try to disconnect the graph. This strategy does *not* lead to any reasonable algorithm, because the graph would get shattered only in the subproblem corresponding to success branches for all $x \in X$. However, there is an intuition that elements of X are reasonable candidates for *branching pivots*: vertices such that branching on them leads to a significant progress of the algorithm.

The main idea presented in [24] is to perform branching while measuring the progress in disconnecting the graph in an indirect way. Let G be the currently considered graph. For a pair of vertices u and v, let the *bucket* of u and v be defined as:

$$\mathcal{B}^G_{u,v} := \{P : P \text{ is an induced path in } G \text{ with endpoints } u \text{ and } v\}.$$

Observe that since G is P_t-free, every element of $\mathcal{B}^G_{u,v}$ is a path on fewer than t vertices, hence $\mathcal{B}^G_{u,v}$ has always at most n^{t-1} elements and can be computed in polynomial time (for a fixed t). On the other hand, $\mathcal{B}^G_{u,v}$ is nonempty if and only if u and v are in the same connected component of G.

Let X be a set whose existence is asserted by Theorem 2.1. Observe that if u and v are in different components of $G - N[X]$, then *all* the paths of $\mathcal{B}^G_{u,v}$ are intersected by $N[X]$. Moreover, as every connected component of $G - N[X]$ has at most $n/2$ elements, this happens for at least half of the pairs $\{u, v\} \in \binom{V(G)}{2}$. Since X has only at most t vertices, by a simple averaging argument we conclude the following.

Claim 1. *There is a vertex x such that $N[x]$ intersects at least a $\frac{1}{t}$ fraction of paths in at least $\frac{1}{2t}$ fraction of buckets.*

A vertex x having the property mentioned in Claim 1 shall be called $\frac{1}{2t}$*-heavy*, or just *heavy*. Then Claim 1 asserts that there is always a heavy vertex; note that such a vertex can be found in polynomial time by inspecting the vertices of G one by one.

We may now present the algorithm:

(1) If G is disconnected, then apply the algorithm to every connected component of G separately.
(2) Otherwise, find a heavy vertex in G and branch on it.

We now sketch a proof of the following claim: on each root-to-leaf path in the recursion tree, this algorithm may execute only $O(\log^2 n)$ success branches. By Claim 1, in each success branch a constant fraction of buckets get their sizes reduced by a constant multiplicative factor. Since buckets are of polynomial size in the first place, after $\Omega(\log n)$ success branches a $\frac{1}{10}$ fraction of the initial buckets must become empty. Since in a connected graph all the buckets are nonempty, it follows that after $\Omega(\log n)$ success branches, the vertex count of the connected graph we are working

on must have decreased by at least a multiplicative factor of 0.01 with respect to the initial graph. As this can happen only $O(\log n)$ times, the claim follows.

Now the recursion tree has depth at most n and each root-to-leaf path contains at most $O(\log^2 n)$ success branches. Therefore, the total size of the recursion tree is $n^{O(\log^2 n)}$, which implies the same bound on the running time. This concludes the description of the algorithm for P_t-free graphs; let us recall that this algorithm was already presented in [24].

2.2 Lifting the Technique to $C_{>t}$-Free Graphs

We now explain how to lift the technique presented in the previous section to the setting of $C_{>t}$-free graphs. As we mentioned before, the main ingredient — the balanced separator theorem — remains true.

Theorem 2.2 (Gyárfás [17], Chudnovsky et al. [8]). *Let G be an n-vertex $C_{>t}$-free graph. Then there is a set X consisting of at most t vertices of G such that $G[X]$ is connected and every connected component of $G - N[X]$ has at most $n/2$ vertices. Furthermore, such a set can be found in polynomial time.*

However, in the previous section we used the P_t-freeness of the graph in question also in one other place: to argue that the buckets $\mathcal{B}^G_{u,v}$ are of polynomial size. This was crucial for the argument that $\Omega(\log n)$ success branches on heavy vertices lead to emptying a significant fraction of the buckets. Solving this issue requires reworking the concept of buckets.

The idea is that in the $C_{>t}$-free case, the objects placed in buckets will connect triples of vertices, rather than pairs. Formally, a *connector* is a graph formed from three disjoint paths Q_1, Q_2, Q_3 by picking one endpoint a_i of Q_i, for each $i = 1, 2, 3$, and either identifying vertices a_1, a_2, a_3 into one vertex, or turning a_1, a_2, a_3 into a triangle; see Figure 1. The paths Q_i are the *legs* of the connector, the other endpoints of the legs are the *tips*, and the (identified or not) vertices a_1, a_2, a_3 are the *center* of the connector. We remark that we allow the degenerate case when one or more paths Q_1, Q_2, Q_3 has only one vertex, but we require the tips to be pairwise distinct.

The following claim is easy to prove by considering any inclusion-wise minimal connected induced subgraph containing u, v, w.

Claim 2. *If vertices u, v, w belong to the same connected component of a graph G, then in G there is an induced connector with tips u, v, w.*

A *tripod* is a connector in which every leg has length at most $t/2+1$ (w.l.o.g. t is even). Every connector contains a *core*: the tripod induced by the vertices at distance at most $t/2$ from the center. The next claim is the key observation that justifies looking at connectors and tripods.

Claim 3. *Let G be a $C_{>t}$-free graph, let T be an induced connector in G, and let X be a subset of vertices such that $G[X]$ is connected and no two tips of T are in the same connected component of $G - N[X]$. Then $N[X]$ intersects the core of T.*

Proof of Claim. Since no two tips of T lie in the same component of $G - N[X]$, it follows that $N[X]$ intersects at least two legs of T, say Q_1 and Q_2 at vertices q_1 and q_2, respectively. We may choose q_1 and q_2 among $N[X] \cap V(Q_1)$ and $N[X] \cap V(Q_2)$ so that they are as close in T as possible to the center of T. Since $G[X]$ is connected, there exists a path P with endpoints q_1 and q_2 such that all the internal vertices of P belong to X. Now P together with the shortest q_1-q_2 path within T form an induced cycle in G. As this cycle must have at most t vertices, we conclude that q_1 or q_2 belongs to the core of T. ◁

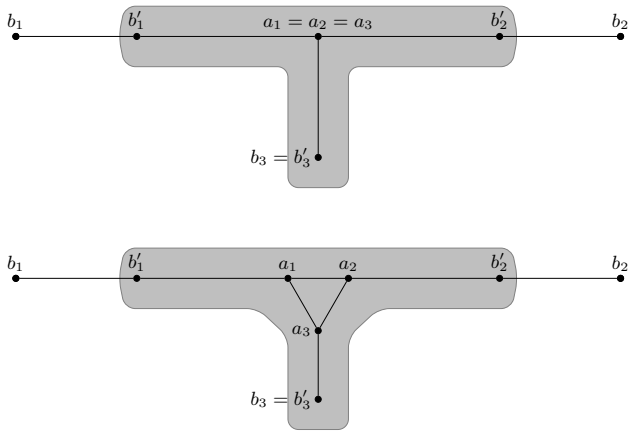

Figure 1: Two connectors with two long legs and one short leg; one connector with a_is identified and one with a_is forming a triangle. Gray outline depicts the core T of the connector (with tips b'_i).

Claim 3 suggests that in $C_{>t}$-free graphs, cores of connectors are objects likely to be hit by balanced separators provided by Theorem 2.2, similarly as in P_t-free graphs, induced paths were likely to be hit by balanced separators given by Theorem 2.1. Let us then use cores as objects for defining buckets.

Let G be a $C_{>t}$-free graph. For an unordered triple $\{u, v, w\} \in \binom{V(G)}{3}$ of distinct vertices, we define the bucket $\mathcal{B}^G_{u,v,w}$ as the set of all cores of all induced connectors with tips u, v, w. Let us stress here that $\mathcal{B}^G_{u,v,w}$ is a set, not a multiset, of tripods: even if some tripod is the core of multiple connectors with tips u, v, w, it is included in $\mathcal{B}^G_{u,v,w}$ only once. Therefore, as each tripod has $O(t)$ vertices, the buckets are again of size $n^{O(t)}$ and can be enumerated in polynomial time. By Claim 2, the bucket $\mathcal{B}^G_{u,v,w}$ is nonempty if and only if u, v, w are in the same connected component of G. Moreover, from Claim 3 we infer the following.

Claim 4. *Let $\{u, v, w\} \in \binom{V(G)}{3}$ be a triple of vertices of G and let X be a vertex subset such that $G[X]$ is connected and no two vertices out of u, v, w belong to the same connected component of $G - N[X]$. Then $N[X]$ intersects all the tripods in the bucket $\mathcal{B}^G_{u,v,w}$.*

Now we would like to obtain an analogue of Claim 1, that is, find a vertex x such that $N[x]$ intersects a significant fraction of tripods in a significant fraction of buckets. Let then X be a set provided by Theorem 2.2 for G. For a moment, let us assume optimistically that each connected component of $G - N[X]$ contains at most $n/10$ vertices, instead of $n/2$ as promised by Theorem 2.2. Observe that if we choose a triple of distinct vertices uniformly at random, then with probability at least $\frac{1}{2}$ no two of these vertices will lie in the same connected component of $G - N[X]$. By Claim 3, this implies

that $N[X]$ intersects all the tripods in at least half of the buckets. By the same averaging argument as before, we get the following.

Claim 5. *Suppose that in G there is a set X consisting of at most t vertices such that $G[X]$ is connected and every connected component of $G - N[X]$ has at most $n/10$ vertices. Then there is a heavy vertex in G.*

Here, we define a heavy vertex as before: it is a vertex x such that $N[x]$ intersects at least a $\frac{1}{t}$ fraction of tripods in at least a $\frac{1}{2t}$ fraction of buckets.

Unfortunately, our assumption that every component of $G - N[X]$ contains at most $n/10$ vertices, instead of at most $n/2$ vertices, is too optimistic. Consider the following example: G is a path on n vertices. The cores of connectors degenerate to subpaths consisting of at most t consecutive vertices of the path, and for every vertex x, the set $N[x]$ intersects any tripod in only an $O(t/n)$ fraction of the buckets. Therefore, in this example there is no heavy vertex at all. We need to resort to a different strategy.

Secondary branching. So let us assume that the currently considered graph G is connected and has no heavy vertex — otherwise we may either recurse into connected components or branch on the heavy vertex (detectable in polynomial time). We may even assume that there is no $(10^{-8}/t)$-*heavy* vertex: a vertex x such that $N[x]$ intersects at least a $(10^{-8}/t)$ fraction of tripods in at least a $(10^{-8}/t)$ fraction of buckets. Indeed, branching on such vertices also leads to quasipolynomial running time (with all factors in the analysis appropriately scaled).

Let us fix a set X provided by Theorem 2.2 for G; then $G[X]$ is connected and each connected component of $G - N[X]$ has at most $n/2$ vertices. By Claim 5, there must be some components of $G - N[X]$ that have more than $n/10$ vertices, for otherwise there would be a heavy vertex. Let C be such a component and let us apply Theorem 2.2 again, this time to $G[C]$, obtaining a connected set Y of size at most t such that every connected component of $G[C] - N[Y]$ has at most $|C|/2$ vertices. If the distance between X and Y is small, say at most $10t$, then one can replace X with the union of X, Y, and a shortest path between X and Y, and repeat the argument. The new set X is still of size $O(t)$, so the argument of Claim 5 applies with adjusted constants, and the absence of a heavy vertex gives another component C' with more than $n/10$ vertices. This process can continue only for a constant number of steps. Hence, at some moment we end up with a connected set X of size $O(t)$ such that every connected component of $G - N[X]$ has at most $n/2$ vertices, a connected component C of $G - N[X]$ with more than $n/10$ vertices, a connected set $Y \subseteq C$ of size at most t such that every connected component of $G[C] - N[Y]$ has at most $|C|/2$ vertices and the distance between X and Y is more than $10t$.

The crucial observation now is as follows: there exists exactly one connected component of $G[C] - N[Y]$, call it D_0, that is adjacent to a vertex of $N[X]$. The existence of at least one such component follows from the connectivity of G. If there were two such components, say D_0 and D_1, then one can construct an induced cycle in G by going from X via D_0 to Y and back to X via D_1. This cycle is long since the distance between X and Y is more than $10t$, which contradicts G being $C_{\geqslant t}$-free. Denote $B := C - D_0$. Note that B is connected and $|B| = |C| - |D_0| \geqslant |C| - |C|/2 = |C|/2 \geqslant n/20$.

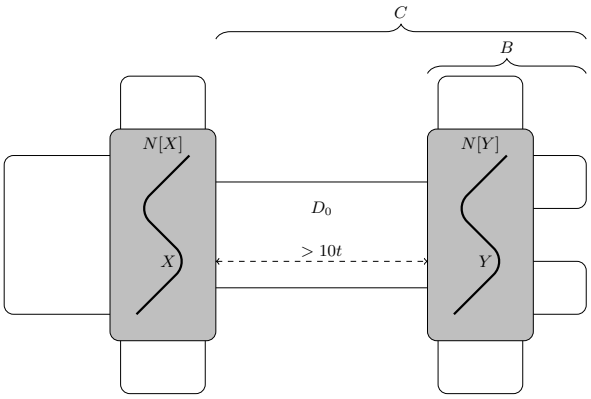

Figure 2: The situation when the secondary branching is invoked.

Repeating the same proof as in the previous observation, note that for every induced subgraph G' of G, there is at most one component of $G'[V(G') \cap C]$ that contains both a vertex of B and a neighbor of $N[X]$: If there were two such components, one could construct a long induced cycle by going from X via the first component to B and back to X via the second one. If such a component exist, we call it *the chip* of G'.

Note that if G' has no chip, then every connected component of G' contains at most $0.95n$ vertices as $n/20 \leqslant |B| \leqslant n/2$. Thus, the goal of the secondary branching is to get to an induced subgraph that contains no chip, that is, to separate B from $N[X]$. The crucial combinatorial insight that we discuss in the next paragraph is that the area of the graph between $N[X]$ and B behaves like a P_t-free graph and is amenable to the branching strategy for P_t-free graphs.

Consider the chip C' in an induced subgraph G' of G. A C'-*link* is a path in G' with endpoints in $N[X] \cap N_{G'}(C')$ and all internal vertices in C'; this path should be induced, except that we allow the existence of an edge between the endpoints. Observe the following:

Claim 6. *Every C'-link has at most t vertices.*

Proof of Claim. Let P be a C'-link. Since the endpoints of P are in $N[X]$ and $G[X]$ is connected, there exists an induced path Q in $G[N[X]]$ with same endpoints as P such that all the internal vertices of P are in X. Then $P \cup Q$ is an induced cycle in G, hence both P and Q must have at most t vertices. ◁

The idea is that in order to cut the chip away, we perform a secondary branching procedure, but this time we use C'-links as objects that are hit by neighborhoods of vertices. Formally, for a pair $\{u, v\} \in \binom{N[X] \cap N_{G'}(C')}{2}$, we consider the *secondary bucket* $\mathcal{L}^{G'}_{u,v}$ consisting of all C'-links with endpoints u and v. Again, by Claim 6, each secondary bucket is of size at most n^t and can be enumerated in polynomial time. Note that $\mathcal{L}^{G'}_{u,v}$ is nonempty for every distinct vertices $u, v \in N_{G'}(C')$.

We shall say that a vertex z of G is *secondary-heavy* if $N[z]$ intersects at least a $\frac{1}{t}$ fraction of links in at least a $\frac{1}{2t}$ fraction of nonempty secondary buckets.

Claim 7. *If $|N_{G'}(C')| \geqslant 2$, then there is a secondary-heavy vertex.*

Proof of Claim (Sketch). We apply a weighted variant of Theorem 2.2 to the graph $G'[N_{G'}[C']]$ in order to find a set $Z \subseteq N_{G'}[C']$ of size at most t such that every connected component of $G'[N_{G'}[C']] - N[Z]$ contains at most half of the vertices of $N_{G'}(C')$. Then $N[Z]$ intersects all the links in at least half of the buckets. The same averaging argument as used before shows that one of vertices of Z is secondary-heavy. ◁

The secondary branching procedure now branches on a secondary-heavy vertex (detectable in polynomial time). This is always possible by Claim 7 as long as $N_{G'}(C')$ contains at least two vertices. If $N_{G'}(C') = \{v\}$ for some vertex v, we choose v as the branching pivot and observe that both in the success and the failure branch there is no chip.

The same analysis as in Section 2.1 shows that branching on secondary-heavy vertices results in a recursion tree with $n^{O(\log^2 n)}$ leaves. In each of these leaves there is no chip, so every connected component of G' contains at most $0.95n$ vertices.

To summarize, we perform branching on $(10^{-8}/t)$-heavy vertices and recursing on connected components as long as a $(10^{-8}/t)$-heavy vertex can be found. When this ceases to be the case, we resort to the secondary branching. Such an application of secondary branching results in producing $n^{O(\log^2 n)}$ subinstances to solve, and in each of these subinstances the size of the largest connected component is at most 95% of the vertex count of the graph for which the secondary branching was initiated. We infer that the running time is $n^{O(\log^3 n)}$. This concludes the description of an $n^{O(\log^3 n)}$-time algorithm for MWIS on $C_{>t}$-free graphs.

2.3 Degeneracy Branching

Our goal in this section is to generalize the approach presented in the previous section to an algorithm solving the following problem: given a vertex-weighted $C_{>t}$-free graph G, find a maximum-weight subset of vertices S such that $G[S]$ is d-degenerate. Here d and t are considered fixed constants. Thus we allow the solution to be just sparse instead of independent, but, compared to Theorem 1.1, so far we do not introduce CMSO_2-expressible properties. Let us call the considered problem Maximum Weight Induced d-Degenerate Graph (MWID).

Recall that a graph G is d-degenerate if every subgraph of G has a vertex of degree at most d. We will rely on the following characterization of degeneracy, which is easy to prove.

Claim 8. *A graph G is d-degenerate if and only if there exists a function $\eta\colon V(G) \to \mathbb{N}$ such that for every $uv \in E(G)$ we have $\eta(u) \neq \eta(v)$ and for each $u \in V(G)$, u has at most d neighbors v with $\eta(v) < \eta(u)$.*

A function $\eta(\cdot)$ satisfying the premise of Claim 8 shall be called a *degeneracy ordering*. Note that we only require that a degeneracy ordering is injective on every edge of the graph, and not necessarily on the whole vertex set. For a vertex u, the value $\eta(u)$ is the *position* of u and the set neighbors of u with smaller positions is the *left neighborhood* of u.

We shall now present a branching algorithm for the MWID problem. For convenience of exposition, let us fix the given $C_{>t}$-free graph G, an optimum solution $S^\star$ in G, and a degeneracy ordering $\eta^\star$ of $G[S^\star]$. We may assume that the co-domain of $\eta^\star$ is $[n] \coloneqq \{1, \ldots, n\}$.

Recall that when performing branching for the MWIS problem, say on a vertex x, in the failure branch we were removing x from the graph, while in the success branch we were removing both x and its neighbors. When working with MWID, we cannot proceed in the same way in the second case, because the neighbors of x can be still included in the solution. Therefore, instead of modifying the graph G along the recursion, we keep track of two disjoint sets A and W: A consists of vertices already decided to be included in the solution, while W is the set of vertices that are still allowed to be taken to the solution in further steps. Initially, $A = \emptyset$ and $W = V(G)$. We shall always branch on a vertex $x \in W$: in the failure branch we remove x from W, while in the success branch we move x from W to A. The intuition is that moving x to A puts more restrictions on the neighbors of x that are still in W. This is because they are now adjacent to one more vertex in A, and they cannot be adjacent to too many, at least as far as vertices with smaller positions are concerned.

For the positions, during branching we will maintain the following two pieces of information:

- a function $\eta\colon A \to [n]$ that is our guess on the restriction of $\eta^\star$ to A; and
- a function $\zeta\colon W \to [n]$ which signifies a *lower bound* on the position of each vertex of W, assuming it is to be included in the solution.

Initially, we set $\zeta(v) = 1$ for each $v \in V(G)$. The quadruple (A, W, η, ζ) as above describes a subproblem solved during the recursion. We will say that such a subproblem is *lucky* if all the choices made so far are compliant with $S^\star$ and $\eta^\star$, that is,

$$A \subseteq S^\star \subseteq A \cup W,$$
$$\eta = \eta^\star|_A,$$
$$\text{and } \eta^\star(u) \geqslant \zeta(u) \text{ for each } u \in S^\star \cap W.$$

Additionally to the above, from a lucky subproblem we also require the following property:

$$\text{for each } v \in A \text{ and } u \in N(v) \cap W \text{ such that } \zeta(u) \leqslant \eta^\star(v), \quad \text{we have } u \in S^\star \text{ and } \eta^\star(u) < \eta^\star(v). \tag{1}$$

In other words, all the neighbors of a vertex $v \in A$ should have their lower bounds larger than the guessed position of v, unless they will be actually included in the solution at positions smaller than that of v. The significance of this property will become clear in a moment.

First, observe that if $G[W]$ is disconnected, then we can treat the different connected components of $G[W]$ separately: for each component D of $G[W]$ we solve the subproblem $(A, D, \eta, \zeta|_D)$ obtaining a solution S_D, and we return $\bigcup_D S_D$ as the solution to (A, W, η, ζ). Property (1) is used to guarantee the correctness of this step: it implies that when taking the union of solutions S_D, the vertices of A do not end up with too many left neighbors.

Thus, we may assume that $G[W]$ is connected. In this case we execute branching on a vertex of W. For the choice of the branching pivot x we use exactly the same strategy as described in the previous section: having defined the buckets in exactly the same way, we always pick x to be a heavy vertex in $G[W]$, or resort to secondary

branching in $G[W]$ (which picks secondary-heavy pivots) in the absence of heavy vertices.

An important observation is that in the success branch — when the vertex $x \in W$ is moved to A — the algorithm notes a significant progress that allows room for additional guessing (by branching). More precisely, on every root-to-leaf path in the recursion tree there are only $O(\log^3 n)$ success branches, which means that following each success branch we can branch further into $n^{O(1)}$ options, and the size of the recursion tree will be still $n^{O(\log^3 n)}$. We use this power to guess (by branching) the following objects when deciding that x should be included in the solution $S^\star$ (here, we assume that the current subproblem is lucky):

- the position $\eta^\star(x)$;
- the set of left neighbors
$$L = \{v \in W \cap N(x) \mid \eta^\star(v) < \eta^\star(x)\}$$
- the positions $(\eta^\star(v) \colon v \in L)$; and
- for each $v \in L$, its left neighbors
$$L_v \coloneqq \{u \in W \cap N(v) \mid \eta^\star(u) < \eta^\star(v)\}.$$

This guess is reflected by the following clean-up operations in the subproblem:

- Move $\{x\} \cup L$ from W to A and set their positions in $\eta(\cdot)$ as the guess prescribes. Note that the vertices of $\bigcup_{v \in L} L_v$ are *not* being moved to A.
- For each $w \in (N(x) \cap W) - L$, increase $\zeta(w)$ to $\max(\zeta(w), \eta(x) + 1)$.
- For each $v \in L$ and $w \in (N(v) \cap W) - L_v$, increase $\zeta(w)$ to $\max(\zeta(w), \eta(v) + 1)$.

It is easy to see that if (A, W, η, ζ) was lucky, then at least one of the guesses leads to considering a lucky subproblem. In particular, property (1) is satisfied in this subproblem. This completes the description of a branching step.

It remains to argue why it is still true that on every root-to-leaf path in the recursion tree there are at most $O(\log^3 n)$ success branches. Before, the key argument was that when a success branch is executed, a constant fraction of buckets (either primary or secondary) loses a constant fraction of elements. Now, the progress is explained by the following claim, which follows easily from the way we perform branching.

Claim 9. *Suppose (A, W, η, ζ) is a lucky subproblem in which branching on x is executed, and let (A', W', η', ζ') be any of the obtained child subproblems. Then for every $y \in N(x) \cap W$, we either have*

$$y \notin W' \qquad \text{or}$$
$$|\{z \in A \cap N(y) \mid \eta(z) < \zeta(y)\}| < |\{z \in A' \cap N(y) \mid \eta'(z) < \zeta'(y)\}|.$$

Note that for $y \in W$, if y gets included in the solution, then the whole set
$$M_y \coloneqq \{z \in A \cap N(y) \mid \eta(z) < \zeta(y)\}$$
must become the left neighbors of y. So if the size of M_y exceeds d, then we can conclude that y cannot be included in the solution and we can safely remove y from W. Thus, the increase of the cardinality of M_y for all neighbors y of x that do not get excluded from consideration is the progress achieved by the algorithm.

Formally, we do as follows. Recall that before, we measured the progress in emptying a bucket $\mathcal{B}^G_{u,v,w}$ by monitoring its size. Now, we monitor the *potential* of $\mathcal{B}^G_{u,v,w}$ defined as
$$\Phi(\mathcal{B}^G_{u,v,w}) \coloneqq \sum_{T \in \mathcal{B}^G(u,v,w)} \sum_{y \in V(T)} (d - |M_y|).$$
Thus, $\Phi(\mathcal{B}^G(u,v,w))$ measures how much the vertices of tripods of $\mathcal{B}^G_{u,v,w}$ have left till saturating their "quotas" for the number of left neighbors. From Claim 9 it can be easily inferred that when branching on a heavy vertex, a constant fraction of buckets lose a constant fraction of their potential, and the same complexity analysis as before goes through.

2.4 CMSO_2 Properties

We now extend the approach presented in the previous section to a sketch of a proof of Theorem 1.1 in full generality. That is, we also take into account CMSO_2-expressible properties.

Degeneracy and treewidth. The first step is to argue that degeneracy and treewidth are functionally equivalent in $C_{>t}$-free graphs, i.e., to prove Theorem 1.3. This part of the reasoning is presented in Section 3.

The argument goes roughly as follows. Suppose, for contradiction, that G is a $C_{>t}$-free d-degenerate graph that has huge treewidth (in terms of d and t). Using known results [18], in G we can find a huge *bramble* $\mathcal{B}$ — a family of connected subgraphs that pairwise either intersect or are adjacent — such that every vertex of G is in at most two elements of $\mathcal{B}$. This property means that $\mathcal{B}$ gives rise to a huge clique minor in G', the graph obtained from G by adding a copy of every vertex (the copy is a true twin of the original). Note that G' is still $C_{>t}$-free and is $2d+1$-degenerate. Now, we can easily prove that the obtained clique minor in G' can be assumed to have *depth* at most t: every branch set induces a subgraph of radius at most t. Using known facts about bounded-depth minors [25, Lemma 2.19 and Corollary 2.20], it follows that G' contains a topological minor model of a large clique that has depth at most $3t+1$: every path representing an edge has length at most $6t+3$. Finally, we show that if we pick at random $t+1$ roots $v_0, \ldots, v_t$ of this topological minor model, and we connect them in order into a cycle in G' using the paths from the model, then with high probability this cycle will be induced. This is because G' is $(2d+1)$-degenerate, so two paths of the model chosen uniformly at random are with high probability nonadjacent, due to their shortness. Thus, we uncovered an induced cycle on more than t vertices in G', a contradiction.

Boundaried graphs and types. We proceed to the proof of Theorem 1.1. By Theorem 1.3, the subgraph $G[S]$ induced by the solution has treewidth smaller than k, where k is a constant that depends only on d and t. Therefore, we will use known compositionality properties of CMSO_2 logic on graphs of bounded treewidth.

For an integer ℓ, an *ℓ-boundaried graph* is a pair (H, ι), where H is a graph and ι is an injective partial function from $V(H)$ to $[\ell]$, called the *labelling*. The domain of ι is the *boundary* of (H, ι) and if $\iota(u) = \alpha$, then u is a boundary vertex with label α. On ℓ-boundaried graphs we have two natural operations: *forgetting* a label — removing a vertex with this label from the domain of ι —

and *gluing* two boundaried graphs — taking their disjoint union and fusing boundary vertices with the same labels. It is not hard to see that a graph has treewidth less than ℓ if and only if it can be constructed from two-vertex ℓ-boundaried graphs by means of these operations.

The crucial, well-known fact about CMSO_2 is that this logic behaves in a compositional way under the operations on boundaried graphs. Precisely, for each fixed ℓ and CMSO_2 sentence φ there is a finite set Types of *types* such that to every ℓ-boundaried graph (H, ι) we can assign $\mathrm{type}(H, \iota) \in \mathrm{Types}$ so that:

- Whether $H \models \varphi$ can be uniquely determined by examining $\mathrm{type}(H, \iota)$.
- The type of the result of gluing two ℓ-boundaried graphs depends only on the types of those graphs.
- The type of the result of forgetting a label in an ℓ-boundaried graph depends only on the label in question and the type of this graph.

In our proof we will use $\ell := 6k$, that is, the boundaries will by a bit larger than the promised bound on the treewidth.

Enriching branching with types. We now sketch how to enrich the algorithm from the previous section to the final branching procedure.The idea is that we perform branching as in the previous section (with significant augmentations, as will be described in a moment), but in order to make sure that the constructed induced subgraph $G[S]$ satisfies φ, we enrich each subproblem with the following information:

- A rooted tree decomposition (T, β) of $G[A]$ of width at most ℓ ($\beta\colon V(T) \to A$ is the bag function).
- For each node a of T, a *projected type* $\mathrm{type}_a \in \mathrm{Types}$.

Again, we fix some optimum solution $S^\star$ together with a d-degeneracy ordering $\eta^\star$ of $G[S^\star]$. Compared to the approach of the previous section, we extend the definition of a subproblem being lucky as follows:

- For each connected component D of $G[W \cap S^\star]$, we require that $N(D) \cap A$ is a set of size at most $4k$ such that there exists a bag of (T, β) that entirely contains it. For such a component D, let $a(D)$ be the topmost node of T satisfying $N(D) \cap A \subseteq \beta(a(D))$.
- For each node a of T, consider the graph H_a induced by $\beta(a)$ and the union of all those components D of $G[W \cap S^\star]$ for which $a(D) = a$. Then the type of H_a with $\beta(a)$ as the boundary is equal to type_a.

Thus, one can imagine the solution $S^\star$ as A plus several extensions into W — vertex sets of components of $G[W \cap S^\star]$. Each such extension D is hanged under a single node $a(D)$ of (T, β) and is attached to it through a neighborhood of size at most $4k$. For each node a of T, we store the projected combined type type_a of all the extensions D for which a is the topmost node to which D attaches. Note that since the algorithm will always make only $\log^{O(1)} n$ success branches along each root-to-leaf path, we maintain the invariant that $|A| \leqslant \log^{O(1)} n$, which implies the same bound on the number of nodes of T.

Recall that in the algorithm presented in the previous section, two basic operations were performed: (a) recursing on connected components of $G[W]$ once this graph becomes disconnected; and (b) branching on a node $x \in W$.

Lifting (a) to the new setting is conceptually simple. Namely, each graph H_a is correspondingly split among the components of $G[W]$, so we guess the projected types of those parts of H_a so that they compose to type_a. The caveat is that in order to make the time complexity analysis sound, we can perform such guessing only when a significant progress is achieved by the algorithm. This is done by performing (a) *only* when each connected component of $G[W]$ contains at most 99% of all the vertices of W, which means that the number of active vertices after this step will drop by 1% in each branch. This requires technical care.

More substantial changes have to be applied to lift operation (b), branching on a node $x \in W$. Failure branch works the same way as before: we just remove x from W. As for success branches, recall that in each of them together with x we move to A the whole set L of left neighbors of x in W. Clearly, the vertices of $L \cup \{x\}$ belong to the same component of $G[A \cap S^\star]$, say D. It would be natural to reflect the move of $L \cup \{x\}$ to A in the decomposition (T, β) as follows: create a new node b with $\beta(b) = (L \cup \{x\}) \cup (N(D) \cap A)$ and make it a child of $a(D)$ in T. Note that thus, $|\beta(b)| \leqslant d + 1 + 4k \leqslant 5k$, so the bound on the width of (T, β) is maintained (even with a margin). However, there is a problem: if by A' we denote the updated A, i.e., $A' = A \cup (L \cup \{x\})$, then the removal of $L \cup \{x\}$ breaks D into several connected components whose neighborhoods in A' are contained in $(N(D) \cap A) \cup (L \cup \{x\})$. This set, however, may have size as large as $4k + d + 1$, so we do not maintain the invariant that every connected component of $G[W \cap S^\star]$ has at most $4k$ neighbors in A.

We remedy this issue using a trick that is standard in the analysis of bounded-treewidth graphs. Since the graph $G[D \cup (N(D) \cap W)]$ has treewidth smaller than k, there exists a set K consisting of at most k vertices of $D \cup (N(D) \cap W)$ such that every connected component of $D - K$ contains at most $|N(D) \cap W|/2$ vertices of $N(D) \cap W$.The algorithm guesses K along with L, moves K to W along with x and L, and and sets $\beta(b) := (N(D) \cap A) \cup (L \cup \{x\}) \cup K$; thus $|\beta(b)| \leqslant 4k + (d+1) + k \leqslant 6k$. Now it is easy to see that due to the inclusion of K, every connected component of $D - (K \cup L \cup \{x\})$ has only at most $2k + k + (d+1) \leqslant 4k$ neighbors in $\beta(b)$, and the problematic invariant is maintained.

This concludes the overview of the proof of Theorem 1.1.

3 $C_{>t}$-FREE GRAPHS OF BOUNDED DEGENERACY HAVE BOUNDED TREEWIDTH

It is well known that if a graph G has treewidth k, then its degeneracy it at most k. However, these parameters can be arbitrarily far away from each other: for instance, 3-regular expanders have degeneracy 3 and treewidth linear in the number of vertices [15]. In this section we prove that if we restrict our attention to $C_{>t}$-free graphs, the treewidth is bounded by a function of degeneracy. In particular, we show Theorem 1.3.

Theorem 1.3. *For every pair of integers d and t, there exists an integer $k = (dt)^{O(t)}$ such that every $C_{>t}$-free graph of degeneracy at most d has treewidth at most k.*

Before we proceed to the proof of Theorem 1.3, let us recall the notion of brambles. Recall that two sets A, b are adjacent if either $A \cap B \neq \emptyset$ or there is an edge with one endpoint in A and the other

in B. For brevity, we say that a set A is adjacent to a vertex v if A is adjacent to $\{v\}$, i.e., either $v \in A$ or v is adjacent to some vertex of A. A *bramble* of size p in a graph G is a collection $\mathcal{B} = (B_1, B_2, \ldots, B_p)$ of nonempty vertex subsets such that each B_i induces a connected graph and all B_is are pairwise adjacent. The sets B_i are called *branch sets*. The *order* of a bramble $\mathcal{B}$ is the size of a smallest set of vertices that hits all branch sets. Observe that the size of a bramble is always at least its order. We will use the following result of Hatzel *et al.* [18], which in a graph of large treewidth constructs a bramble of large order in which no vertex participates in more than two branch sets.

Theorem 3.1 (Hatzel *et al.* [18]). *There exists a polynomial* $\mathrm{p}(\cdot)$ *such that for every positive integer* k*, every graph* G *of treewidth at least* k *contains a bramble* $\mathcal{B}$ *of order at least* $\sqrt{k}/\mathrm{p}(\log k)$ *such that each vertex of* G *is in at most two branch sets of* $\mathcal{B}$*.*

We now proceed to the proof of Theorem 1.3. Without loss of generality we may assume that t is even, $t \geqslant 4$, and $d \geqslant 2$. For contradiction, suppose that G is a $C_{>t}$-free graph with degeneracy at most d and treewidth larger that

$$k := \left(500\,000 \cdot d^2 t^5\right)^{4t+4} \cdot \left[\mathrm{p}\left(\log\left((500\,000 \cdot d^2 t^5)^{4t+4}\right)\right)\right]^4,$$

where $\mathrm{p}(\cdot)$ is the polynomial provided by Theorem 3.1. Thus, by applying Theorem 3.1 to G we obtain a bramble $\mathcal{B} = (B_1, B_2, \ldots, B_p)$ of order

$$p > \frac{\sqrt{k}}{\mathrm{p}(\log k)} \geqslant \left(500\,000 \cdot d^2 t^5\right)^{2t+2}.$$

Note that we can assume that each branch set of $\mathcal{B}$ is inclusion-wise minimal (subject to $\mathcal{B}$ being a bramble), as otherwise we can remove some vertices from branch sets. Therefore, for each branch set B_i and each vertex v of B_i, either there is some branch set B_j which is adjacent to v but nonadjacent to $B_i - \{v\}$, or v is a cutvertex in $G[B_i]$ and its role is to keep the branch set connected.

Claim 10. *For each* $i \in [p]$*, and all* $u, v \in B_i$*, the distance between* u *and* v *in* $G[B_i]$ *is at most* t*.*

Proof of Claim. For contradiction, suppose that there is B_i violating the claim. Let u, v be the vertices at maximum distance in $G[B_i]$, by assumption this distance is at least $t+1$. As u and v are the ends of a maximal path in $G[B_i]$, none of them is a cutvertex in $G[B_i]$. Thus there is a branch set B_u which is adjacent only to u in B_i, and another branch set which is adjacent only to v in B_i. Recall that $B_u \cup B_v$ is connected and nonadjacent to $B_i - \{u, v\}$. So by concatenating a shortest u-v-path in B_i and a shortest u-v-path in $B_u \cup B_v$, we obtain an induced cycle with at least $t+1$ vertices, a contradiction. ◁

Let G' be the lexicographic product $G \bullet K_2$: the graph obtained from G by introducing, for each $x \in V(G)$, a copy x' of x and making it adjacent to x, all neighbors of x, and all their copies. Note that in G', the copy x' is a true twin of x. Observe also that the degeneracy of G' is at most $2d+1$: we can modify a d-degeneracy ordering of G into a $(2d+1)$-degeneracy ordering of G' by inserting each vertex x' immediately after x.

Claim 11. *The graph* G' *contains* K_p *as a depth-*t *minor.*

Proof of Claim. We construct a family $\mathcal{B}' = (B'_1, B'_2, \ldots, B'_p)$ as follows. We start with $B'_i := B_i$ for all $i \in [p]$ and we iteratively inspect every vertex x of G. If x belongs to more than one of the sets $\{B_1, \ldots, B_p\}$, then, by the properties given by Theorem 3.1, x must belong to exactly two of them, say $x \in B_i \cap B_j$ for some $i \neq j$. Then replace x with x' in B'_j, thus making B'_i and B'_j not overlap on x.

It is clear that once this operation is applied to each vertex of G, the resulting sets of $\mathcal{B}'$ are pairwise disjoint and pairwise adjacent. Further, for each $i \in [p]$ the graph $G'[B'_i]$ is isomorphic to $G[B_i]$, as we only replaced some vertices by their true twins, so in particular $G'[B'_i]$ is connected. Therefore, $\mathcal{B}'$ is a minor model of a clique of order p in G'. By Claim 10, the radius of each graph $G'[B'_i]$ is at most t, hence this model has depth at most t. ◁

The next result binds the maximum size of a bounded-depth clique minor and the maximum size of a bounded-depth topological clique minor that can be found in a graph. It is a fairly standard fact used in the sparsity theory; for the proof, see e.g. [25, Lemma 2.19 and Corollary 2.20].

Proposition 3.2. *Let* G *be a graph and let* t, p, p' *be integers such that* $p \geqslant 1 + (p'+1)^{2t+2}$*. If* G *contains* K_p *as a depth-*t *minor, then* G *contains* $K_{p'}$ *as a depth-*$(3t+1)$ *topological minor.*

By combining Claim 11 and Proposition 3.2, we conclude that G'' contains $K_{p'}$ as a topological depth-$(3t+1)$ minor, where

$$p' := \left\lfloor \frac{p^{\frac{1}{2t+2}}}{4} \right\rfloor \geqslant 100\,000 \cdot d^2 \cdot t^5.$$

Fix some topological depth-$(3t+1)$ minor model of $K_{p'}$ in G'. Let R be the set of roots of the minor model and consider the graph $G'[R]$. It has p' vertices and, as a subgraph of G', is $(2d+1)$-degenerate. Therefore, there is an independent set R' in $G'[R]$ of size at least

$$p'' := \left\lceil \frac{p'}{2d+2} \right\rceil \geqslant \frac{100\,000 \cdot d^2 \cdot t^5}{2d+2} \geqslant 20\,000 \cdot d \cdot t^5.$$

Observe that restricting our minor model only to the roots that are in R' and paths incident to them gives us a topological depth-$(3t+1)$ minor model of $K_{p''}$ with the additional property that the roots are pairwise nonadjacent.

Let H be the subgraph of G' induced by the vertices used by the topological minor model obtained in the previous step. Let X be the set of vertices of H with degree larger than $200 \cdot d \cdot t^2$, which are not roots. Since H is $(2d+1)$-degenerate, we observe that

$$\begin{aligned} |X| &\leqslant \frac{(2d+1)|V(H)|}{100 \cdot dt^2} \leqslant \frac{(2d+1)(6t+3)\binom{p''}{2}}{100 \cdot dt^2} \leqslant \frac{20}{100t}\binom{p''}{2} \\ &= \varepsilon \cdot \binom{p''}{2}, \text{ where } \varepsilon := \frac{1}{5t}. \end{aligned}$$

Let H' be obtained from H by removing all vertices in X, along with all paths from the topological minor model which contain a vertex from X. Note that thus, we have removed at most $\varepsilon \cdot \binom{p''}{2}$ paths.

Observe that H' still contains a depth-$(3t+1)$ topological minor model of some graph Z with p'' vertices and at least

$$\binom{p''}{2} - |X| \geqslant \binom{p''}{2} - \varepsilon\binom{p''}{2} = (1-\varepsilon)\binom{p''}{2}$$

edges. Thus, the average degree of a vertex in Z is at least $(1-\varepsilon)(p''-1)$.

Let $\mathcal{W} = (v_0, v_1, \ldots, v_{t/2})$ be a sequence of vertices of Z, chosen independently and uniformly at random. In what follows, all arithmetic operations on the indices of the vertices v_i are computed modulo $t/2+1$, in particular $v_{t/2+1} = v_0$.

We prove that with positive probability, $\mathcal{W}$ has the following four properties:

(P1) The vertices v_i are pairwise distinct.
(P2) For every $0 \leqslant i \leqslant t/2$, v_iv_{i+1} is an edge of Z; let P_i be the corresponding path in H'.
(P3) For every $0 \leqslant i \leqslant t/2$ and $0 \leqslant j \leqslant t/2$ such that $j \notin \{i, i+1\}$, the internal vertices on the path P_i are anti-adjacent to v_j.
(P4) For all $0 \leqslant i < j \leqslant t/2$, the internal vertices of P_i are anti-adjacent to the internal vertices of P_j.

Observe that these four properties imply that the concatenation of all paths P_i is a hole of length more than t in G' (recall here that the roots of the minor model are independent in H'). The assumption that G is $C_{>t}$-free implies that G' is $C_{>t}$-free as well, hence this will be a contradiction.

For (P1), since $p'' \geqslant 20\,000 \cdot d \cdot t^5$, by the union bound the probability that $v_i = v_j$ for some $i \neq j$ is at most $\binom{t}{2}/p'' < 0.1$.

For (P2), since v_i and v_{i+1} are independently chosen vertices, and Z has at least $(1-\varepsilon)\binom{p''}{2}$ edges, the probability that v_iv_{i+1} is not an edge of Z is bounded by $\varepsilon = \frac{1}{5t}$. By the union bound, the probability that (P2) does not hold is bounded by $\varepsilon \cdot (t/2+1) \leqslant 0.2$.

For (P3), fix $0 \leqslant i \leqslant t/2$ and assume $v_iv_{i+1} \in E(Z)$ so that P_i is defined. Then, the total number of neighbors of the internal vertices of P_i is bounded by $(6t+3) \cdot 200 \cdot d \cdot t^2 \leqslant 2000 \cdot d \cdot t^3$. Since v_j is a vertex of $V(Z)$ chosen at random independently of the choice of v_i and v_{i+1}, the probability that v_j is among these neighbors is bounded by $2000 \cdot dt^3/p'' \leqslant 0.1/t^2$. By the union bound, (P2) holds but (P3) does not hold with probability at most $t(t-2) \cdot \frac{0.1}{t^2} \leqslant 0.1$.

For (P4), fix $0 \leqslant i < j \leqslant t/2$. Note that it may be possible that $i+1 = j$ or $j+1 = i$ (cyclically modulo $t/2+1$), but not both. Hence, by symmetry between i and j, assume that the choice of v_{j+1} is independent of the choices of v_i, v_{i+1}, and v_j. Assume that $v_iv_{i+1} \in E(Z)$ so that P_i is defined. As in the previous paragraph, there are at most $2000dt^3$ neighbors in H' of the internal vertices of P_i. There are $p'' = |V(Z)|$ choices for v_{j+1}, all of them leading to either $v_jv_{j+1} \notin E(Z)$ or to vertex-disjoint (except for v_j) choices of the path P_j. Hence, for at most $2000dt^3$ of these choices, we have $v_jv_{j+1} \in E(Z)$ but there is an edge between an internal vertex of P_j and an internal vertex of P_i. By the union bound, (P2) holds but (P4) does not hold with probability less than $\binom{t/2+1}{2} \cdot \frac{2000dt^3}{p''} \leqslant \binom{t/2+1}{2} \cdot \frac{2000dt^3}{20\,000dt^5} \leqslant 0.1$.

By the union bound over all the above cases, $\mathcal{W}$ satisfies all properties (P1)–(P4) with probability at least $1-0.1-0.2-0.1-0.1 = 0.5$. This gives the desired contradiction and completes the proof.

4 A SIMPLE TECHNIQUE FOR APPROXIMATION SCHEMES

In this final section we present a simple technique for turning polynomial-time and quasipolynomial-time algorithms for MWIS on P_t-free and $C_{>t}$-free graphs into PTASes and QPTASes for more general problem, definable as looking for the largest induced subgraph that belongs to some weakly hyperfinite class. Let us stress that this technique works only for unweighted problems.

We define the *blob graph* of a graph G, denoted G°, as the graph defined as follows:

$$V(G^\circ) := \{X \subseteq V(G) \mid G[X] \text{ is connected}\},$$
$$E(G^\circ) := \{X_1X_2 \mid X_1 \text{ and } X_2 \text{ are adjacent}\}.$$

The main combinatorial insight of this section is the following combinatorial property of G°. Let us point out that a similar result could be derived from the work of Cameron and Hell [6], although it is not stated there explicitly.

Theorem 4.1. *Let G be a graph. The following hold.*

(S1) The length of a longest induced path in G° is equal to the length of a longest induced path in G.
(S2) The length of a longest induced cycle in G° is equal to the length of a longest induced cycle in G, with the exception that if G has no cycle at all (G is a forest), then G° may contain triangles, but it has no induced cycles of length larger than 3 (i.e. it is a chordal graph).

Proof. Note that since G is an induced subgraph of G° (as witnessed by the mapping $u \mapsto \{u\}$), we only need to upper-bound the length of a longest induced path (resp., cycle) in G° by the length of a longest induced path (resp. cycle) in G.

Let $P^\circ = X_1, X_2, \ldots, X_t$ be an induced path in G°. We observe that the graph $G[\bigcup_{j=1}^{t} X_j]$ is connected and for each $j' \in [t-2]$ the sets $\bigcup_{j=1}^{j'} X_j$ and $\bigcup_{j=j'+2}^{t} X_j$ are nonadjacent.

Fix an induced path $P = v_1, v_2, \ldots, v_p$ in $G[\bigcup_{i=1}^{t} X_i]$. We define

$$\mathrm{set}(i) := \max\{j \mid \{v_1, v_2, \ldots, v_i\} \cap X_j \neq \emptyset\}.$$

Claim 12. *For all $i \in [p-1]$ it holds that $\mathrm{set}(i+1) \in \{\mathrm{set}(i), \mathrm{set}(i)+1\}$.*

Proof of Claim. It is clear that $\mathrm{set}(i+1) \geqslant \mathrm{set}(i)$, so suppose $\mathrm{set}(i+1) \geqslant \mathrm{set}(i)+2$. Since v_iv_{i+1} is an edge of G, we conclude that there is an edge in G° between the sets $\{X_j \mid j \leqslant \mathrm{set}(i)\}$ and $\{X_j \mid j \geqslant \mathrm{set}(i)+2\}$, a contradiction with P° being induced. ◁

The following claim encapsulates the main idea of the proof.

Claim 13. *Let $P^\circ = X_1, X_2, \ldots, X_t$ be an induced path in G° such that $X_1 \not\subseteq X_2$. Let $X_1' \subseteq X_1 - X_2$ and $X_t' \subseteq X_t$ be nonempty sets. Let $P = v_1, v_2, \ldots, v_p$ be a shortest path in $G[\bigcup_{j=1}^{t} X_j]$ such that $v_1 \in X_1'$ and $v_p \in X_t'$. Then P is induced, $p \geqslant t$, and $\{v_2, v_3, \ldots, v_{p-1}\} \cap (X_1' \cup X_t') = \emptyset$.*

Proof of Claim. The path P is induced and $\{v_2, v_3, \ldots, v_{p-1}\} \cap (X_1' \cup X_t') = \emptyset$ by the minimality assumption. Recall that X_1 must be disjoint with $\bigcup_{j=3}^{t} X_j$. Thus $\mathrm{set}(1) = 1$ and $\mathrm{set}(p) = t$, so the claim follows from Claim 12. ◁

Now we are ready to prove (S1). Our goal is to prove that if G° contains an induced path on t vertices, then so does G. If $t = 1$, then the statement is trivial, so assume that $t \geqslant 2$ and let $P^\circ = X_1, X_2, \ldots, X_t$ be an induced path in G°.

If $X_1 \not\subseteq X_2$, then we are done by Claim 13 applied to P° for $X_1' = X_1 - X_2$ and $X_t' = X_t$. So assume that $X_1 \subseteq X_2$ and note that $X_2 \not\subseteq X_1$, for X_1 and X_2 are two different vertices of P°. If $t = 2$,

then any edge from X_1 to $X_2 - X_1$ is an induced path in G with two vertices; such an edge exists as $G[X_2]$ is connected. So from now on we may assume $t \geqslant 3$.

Let $X_2' \subseteq X_2 - X_1$ be such that $G[X_2']$ is a connected component of $G[X_2 - X_1]$ and X_2' and X_3 are adjacent. Such a set exists as X_3 is adjacent to X_2, but nonadjacent to X_1. Note that $G[X_2]$ being connected implies that there exists a nonempty set $X_2'' \subseteq X_2'$, such that every vertex from X_2'' has a neighbor in X_1. Furthermore, $X_2'' \cap X_3 = \emptyset$, as X_1 is nonadjacent to X_3. Observe that $\widehat{P}^\circ := X_2', X_3, \ldots, X_t$ is an induced path in G° with at least $t - 1 \geqslant 2$ vertices, such that $X_2' \not\subseteq X_3$. Let $P' = v_2, v_3, \ldots, v_p$ be the induced path in G with at least $t - 1$ vertices obtained by Claim 13 applied to $\widehat{P}^\circ$, X_2'', and X_t. Now recall that $v_2 \in X_2''$, so there is $v_1 \in X_1$ adjacent to v_2. Note that v_1 is nonadjacent to every v_i for $i > 2$, because $v_i \notin X_2''$ for $i > 2$. Thus $P := v_1, v_2, \ldots, v_p$ is an induced path in G with at least t vertices.

Now let us prove (S2). We proceed similarly to the proof of (S1). If G° is chordal (every induced cycle is of length 3), then we are done by the exceptional case of the statement. Otherwise, let $C^\circ = X_1, X_2, \ldots, X_t$ be an induced cycle in G° for some $t \geqslant 4$; we want to find an induced cycle of length at least t in G. Note that $X_t \not\subseteq X_{t-1}$ and $X_t \not\subseteq X_1$, as otherwise C° is not induced. We observe that there are nonempty sets $X_t^1 \subseteq X_t$ and $X_t^{t-1} \subseteq X_t$, such that every vertex from X_t^1 has a neighbor in X_1 and every vertex from X_t^{t-1} has a neighbor in X_{t-1}. Let Q be a shortest path contained in X_t whose one endvertex, say x^1 is in X_t^1 and the other endvertex, say x^{t-1} is in X_t^{t-1}. Note that it is possible that $x^1 = x^{t-1}$. The minimality of Q implies that no vertex of Q, except for x^1, x^{t-1}, has a neighbor in $\bigcup_{j=1}^{t-1} X_j$.

Let P° be the induced path $X_1, X_2, \ldots, X_{t-1}$. Denote $X_1' := N(x^1) \cap X_1$ and $X_{t-1}' := N(x^{t-1}) \cap X_{t-1}$. Recall that both these sets are nonempty and $X_1' \cap X_2 = \emptyset$ and $X_{t-1}' \cap X_{t-2} = \emptyset$. Let $P = v_1, v_2, \ldots, v_p$ be the induced path given by Claim 13 for P°, X_1', and X_{t-1}'. Recall that $p \geqslant t - 1$. Now let C be the cycle obtained by concatenating P and Q, and observe that the cycle C is induced. Furthermore, as P has at least $t - 1$ vertices and Q has at least one vertex, C has at least t vertices, which completes the proof. □

Let us define an auxiliary problem called Maximum Induced Packing. An instance of Maximum Induced Packing is a triple $(G, \mathcal{F}, \mathfrak{w})$, where G is a graph, $\mathcal{F}$ is a family of connected induced subgraph of G, and $\mathfrak{w} \colon \mathcal{F} \to \mathbb{R}_+$ is a weight function. A *solution* to $(G, \mathcal{F}, \mathfrak{w})$ is a set $X \subseteq V(G)$, such that

- each connected component of $G[X]$ belongs to $\mathcal{F}$; and
- $\sum_{C\colon \text{ component of } G[X]} \mathfrak{w}(C)$ is maximized.

We observe the following.

Theorem 4.2. *Let $(G, \mathcal{F}, \mathfrak{w})$ be an instance of Maximum Induced Packing, where $|\mathcal{F}| = N$.*

(1) If G is P_t-free for some integer t, then the instance $(G, \mathcal{F}, \mathfrak{w})$ can be solved in time $N^{O(\log^2 N)}$.
(2) If G is $C_{>t}$-free for some integer t, then the instance $(G, \mathcal{F}, \mathfrak{w})$ can be solved in time $N^{O(\log^3 N)}$.
(3) If G is P_6-free or $C_{>4}$-free, then the instance $(G, \mathcal{F}, \mathfrak{w})$ can be solved in time $N^{O(1)}$.

Proof. Let G' be the subgraph of G° induced by $\mathcal{F}$. Clearly, G' has N vertices. We observe that solving the instance $(G, \mathcal{F}, \mathfrak{w})$ of Maximum Induced Packing is equivalent to solving the instance $(G', \mathfrak{w})$ of MWIS. Now the theorem follows from Theorem 4.1 and the fact that MWIS can be solved in time $n^{O(\log^2 n)}$ in n-vertex P_t-free graphs [13, 24], in time $n^{O(\log^3 n)}$ in n-vertex $C_{>t}$-free graphs, using Theorem 1.1 only for MWIS, and in polynomial time in P_6-free [16] or $C_{>4}$-free graphs [1]. □

As an example of an application of Theorem 4.2, we obtain the following corollary.

Corollary 4.3. *For every fixed d and t, given an n-vertex P_t-free graph G, in time $n^{O(\log^2 n)}$ we can find the largest induced subgraph of G with maximum degree at most d.*

Proof. Note that every connected P_t-free graph with maximum degree at most d has at most d^t vertices. Thus, the family $\mathcal{F}$ of all connected induced subgraphs of G with maximum degree at most d has size at most $N := n^{d^t}$ and can be enumerated in polynomial time. For each $F \in \mathcal{F}$ set $\mathfrak{w}(F) := |V(F)|$. We may now apply Theorem 4.2 to solve the instance $(G, \mathcal{F}, \mathfrak{w})$ of Maximum Induced Packing in time $N^{O(\log^2 N)} = n^{O(\log^2 n)}$. □

Note that the strategy we used to prove Theorem 4.2 cannot be used to solve Max Induced Forest in quasipolynomial time, as there can be arbitrarily larger P_t-free tree; consider, for instance, the family of stars. However, it is sufficient to obtain a simple QPTAS for the unweighted version of the problem.

A class of graphs C is called *weakly hyperfinite* if for every $\varepsilon > 0$ there is $c(\varepsilon) \in \mathbb{N}$, such that in every graph $F \in C$ there is a subset X of at least $(1 - \varepsilon)|V(F)|$ vertices such that every connected component of $F[X]$ has at most $c(\varepsilon)$ vertices [22, Section 16.2]. Weakly hyperfinite classes are also known under the name *fragmentable* [12]. Every class closed under edge and vertex deletion which has sublinear separators is weakly hyperfinite [22, Theorem 16.5], hence well-known classes of sparse graphs, such as planar graphs, graphs of bounded genus, or in fact all proper minor-closed classes, are weakly hyperfinite.

For a class C of graphs, by Largest Induced C-Graph we denote the following problem: given a graph G, find a largest induced subgraph of G, which belongs to C. To make the problem well defined, we will always assume that $K_1 \in C$. We can now conclude the following.

Theorem 4.4. *Let C be a nonempty, weakly hyperfinite class of graphs, which is closed under vertex deletion and disjoint union operations. Then, the Largest Induced C-Graph problem*

(1) has a QPTAS in $C_{>t}$-free graphs, for every fixed t; and
(2) has a PTAS in P_6-free graphs and in $C_{>4}$-free graphs.

Proof. Let n be the number of vertices of the given graph G and let ε be the desired accuracy, i.e., the goal is to find a solution whose size is at least a $(1 - \varepsilon)$ fraction of the optimum. Let $c := c(\varepsilon)$.

Let X^* be the vertex set of an optimum solution. By the properties of C, there exists $X' \subseteq X^*$ of size at least $(1 - \varepsilon)|X^*|$ such that each connected component of $G[X']$ has at most c vertices. Let $\mathcal{F}$ be the set of all connected induced subgraphs of G that have at most c

vertices and belong to C. Clearly $|\mathcal{F}| \leqslant n^c$ and $\mathcal{F}$ can be enumerated in polynomial time. For each $F \in \mathcal{F}$, we set $\mathfrak{w}(F) := |V(F)|$.

Apply the algorithm of Theorem 4.2 to solve the instance $(G, \mathcal{F}, \mathfrak{w})$ of Maximum Induced Packing in time $n^{O(\log^3 n^c)} = n^{O(\log^3 n)}$ if G is $C_{>t}$-free, or in polynomial time if G is P_6-free or $C_{>4}$-free. Let X be the optimum solution found by the algorithm. As C is closed under the disjoint union operation, we observe that $G[X]$ is a feasible solution to Largest Induced C-Graph. Moreover we have $|X| \geqslant |X'| \geqslant (1-\varepsilon)|X^*|$. □

ACKNOWLEDGMENTS

Peter Gartland and Daniel Lokshtanov were supported by BSF award 2018302 and and NSF award CCF-2008838.

The work of Marcin Pilipczuk and Michał Pilipczuk is a part of projects CUTACOMBS (grant agreement No. 714704) and TOTAL (grant agreement No. 677651) that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme.

Paweł Rzążewski was supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.

REFERENCES

[1] Tara Abrishami, Maria Chudnovsky, Marcin Pilipczuk, Paweł Rzążewski, and Paul D. Seymour. 2021. Induced subgraphs of bounded treewidth and the container method. In *Proceedings of the Fourth SIAM Symposium on Discrete Algorithms (SODA), Alexandria, Virginia, USA, January 10-13, 2021 (Virtual conference)*. SIAM, 1948–1964. https://doi.org/10.1137/1.9781611976465.116

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

[3] Vladimir E. Alekseev. 2004. Polynomial algorithm for finding the largest independent sets in graphs without forks. *Discrete Applied Mathematics* 135, 1–3 (2004), 3–16. https://doi.org/10.1016/S0166-218X(02)00290-1

[4] Aistis Atminas, Vadim V. Lozin, and Igor Razgon. 2012. Linear Time Algorithm for Computing a Small Biclique in Graphs without Long Induced Paths. In *Proceedings of the 13th Scandinavian Symposium and Workshops Algorithm Theory, SWAT 2012 (Lecture Notes in Computer Science, Vol. 7357)*. Springer, 142–152. https://doi.org/10.1007/978-3-642-31155-0_13

[5] Gábor Bacsó, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Zsolt Tuza, and Erik Jan van Leeuwen. 2019. Subexponential-Time Algorithms for Maximum Independent Set in P_t-Free and Broom-Free Graphs. *Algorithmica* 81, 2 (2019), 421–438. https://doi.org/10.1007/s00453-018-0479-5

[6] Kathie Cameron and Pavol Hell. 2006. Independent packings in structured graphs. *Math. Program.* 105, 2-3 (2006), 201–213. https://doi.org/10.1007/s10107-005-0649-5

[7] Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. 2019. Quasi-polynomial time approximation schemes for the Maximum Weight Independent Set Problem in H-free graphs. *CoRR* abs/1907.04585 (2019). arXiv:1907.04585 http://arxiv.org/abs/1907.04585

[8] Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. 2020. Quasi-polynomial time approximation schemes for the Maximum Weight Independent Set Problem in H-free graphs. In *Proceedings of the Thirty-First ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*. SIAM, 2260–2278. https://doi.org/10.1137/1.9781611975994.139

[9] Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.* 85, 1 (1990), 12–75. https://doi.org/10.1016/0890-5401(90)90043-H

[10] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications, Vol. 138. Cambridge University Press. http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR

[11] Guoli Ding. 1992. Subgraphs and well-quasi-ordering. *J. Graph Theory* 16, 5 (1992), 489–502. https://doi.org/10.1002/jgt.3190160509

[12] Keith Edwards and Graham Farr. 2001. Fragmentability of Graphs. *Journal of Combinatorial Theory, Series B* 82, 1 (2001), 30 – 37. https://doi.org/10.1006/jctb.2000.2018

[13] Peter Gartland and Daniel Lokshtanov. 2020. Independent Set on P_k-Free Graphs in Quasi-Polynomial Time. (2020), 613–624. https://doi.org/10.1109/FOCS46700.2020.00063

[14] Peter Gartland, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. 2021. Finding large induced sparse subgraphs in $C_{>t}$-free graphs in quasipolynomial time. *CoRR* abs/2007.11402 (2021). arXiv:2007.11402 https://arxiv.org/abs/2007.11402

[15] Martin Grohe and Dániel Marx. 2009. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B* 99, 1 (2009), 218–228. https://doi.org/10.1016/j.jctb.2008.06.004

[16] Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. 2019. Polynomial-time algorithm for Maximum Weight Independent Set on P_6-free graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*. SIAM, 1257–1271. https://doi.org/10.1137/1.9781611975482.77

[17] András Gyárfás. 1987. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae* 19 (1987), 413–441.

[18] Meike Hatzel, Pawel Komosa, Marcin Pilipczuk, and Manuel Sorge. 2020. Constant Congestion Brambles. *CoRR* abs/2008.02133 (2020). arXiv:2008.02133 https://arxiv.org/abs/2008.02133

[19] Daniel Lokshtanov, Martin Vatshelle, and Yngve Villanger. 2014. Independent Set in P_5-Free Graphs in Polynomial Time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*. SIAM, 570–581. https://doi.org/10.1137/1.9781611973402.43

[20] Vadim V. Lozin and Martin Milanič. 2008. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *J. Discrete Algorithms* 6, 4 (2008), 595–604. https://doi.org/10.1016/j.jda.2008.04.001

[21] George J. Minty. 1980. On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B* 28, 3 (1980), 284–304. https://doi.org/10.1016/0095-8956(80)90074-X

[22] Jaroslav Nešetřil and Patrice Ossona de Mendez. 2012. *Sparsity — Graphs, Structures, and Algorithms*. Algorithms and combinatorics, Vol. 28. Springer. https://doi.org/10.1007/978-3-642-27875-4

[23] Jana Novotná, Karolina Okrasa, Michal Pilipczuk, Paweł Rzążewski, Erik Jan van Leeuwen, and Bartosz Walczak. 2020. Subexponential-Time Algorithms for Finding Large Induced Sparse Subgraphs. *Algorithmica* (July 2020). https://doi.org/10.1007/s00453-020-00745-z

[24] Michał Pilipczuk, Marcin Pilipczuk, and Paweł Rzążewski. 2021. Quasi-polynomial-time algorithm for Independent Set in P_t-free graphs via shrinking the space of induced paths. In *Proceedings of the Fourth SIAM Symposium on Simplicity in Algorithms (SOSA), Alexandria, Virginia, USA, January 11-12, 2021 (Virtual conference)*, Valerie King and Hung Viet Le (Eds.). SIAM, 204–209. https://doi.org/10.1137/1.9781611976496.23

[25] Marcin Pilipczuk, Michał Pilipczuk, and Sebastian Siebertz. Winter semester 2019/20. Lecture notes for the course Sparsity. available at: https://www.mimuw.edu.pl/~mp248287/sparsity2/.

[26] Najiba Sbihi. 1980. Algorithme de recherche d'un stable de cardinalite maximum dans un graphe sans etoile. *Discrete Mathematics* 29, 1 (1980), 53–76. (in French).